Caractériser et détecter les communautés dans les réseaux sociaux

Jean Creusefond

Normandie Université

Thèse de doctorat

Pour obtenir le diplôme de doctorat

Spécialité : Informatique et applications

Préparée au sein de l'Université de Caen Normandie

Caractériser et détecter les communautés dans les réseaux sociaux

Présentée et soutenue par
Jean CREUSEFOND

Thèse soutenue publiquement le 17/01/2016 devant le jury composé de		
Mme. Anne BOYER	Professeur des Universités Université de Nancy 2, LORIA	Rapporteuse
Mr. Christoph DÜRR	Directeur de recherche CNRS Université de Pierre et Marie Curie, LIP6	Examinateur
Mr. Thomas LARGILLIER	Maître de conférences Université de Caen Normandie, GREYC	Examinateur
Mr. Matthieu LATAPY	Directeur de recherche CNRS Université de Pierre et Marie Curie, LIP6	Rapporteur
Mr. Sylvain PEYRONNET	Professeur des Universités Université de Caen Normandie, GREYC	Directeur
Mr. Marc SPANIOL	Professeur des Universités Université de Caen Normandie, GREYC	Examinateur
Mr. Emmanuel VIENNET	Professeur des Universités Université Paris 13, L2TI	Examinateur

Thèse dirigée par Sylvain Peyronnet, laboratoire GREYC

UNICAEN
UNIVERSITÉ
CAEN
NORMANDIE

GREYC

Remerciements

Tout d'abord, j'aimerais remercier Sylvain Peyronnet et Thomas Largillier pour leur encadrement tout au long de cette thèse. Sans leur patience, leur sincérité et leur implication, cette thèse n'aurait jamais vu le jour. Sylvain, Thomas, je vous dois beaucoup.

Je remercie aussi Christoph Dürr et Emmanuel Viennet d'avoir accepté de faire partie de mon jury, et Marc Spaniol de le présider. En plus d'être membres de mon jury, Anne Boyer et Matthieu Latapy m'ont fait l'honneur de rapporter ma thèse, et je les en remercie profondément.

Je souhaite aussi mentionner les interlocuteurs avec lesquels j'ai eu des discussions qui ont étés des points clef lors de ce doctorat. Gaetan Richard, merci pour ta curiosité insatiable et ton entrain à toute épreuve. Loïc Lhote, merci pour ton humour noir et ton enthousiasme. Michel Habib, merci pour tes idées ingénieuses et tes projets innovants. Nicolas Bacquey, merci pour ton regard critique et ton point de vue formel. Marc Spaniol, merci pour tes propositions pertinentes et ton intérêt qui m'a beaucoup encouragé.

J'ai aussi reçu beaucoup d'aide pour louvoyer dans l'administration. Merci à Julien Clément pour son soutien au niveau de l'équipe AMACC, Marie Meleux pour sa patience envers mon cas atypique pour l'école doctorale et Arielle Perette pour sa gestion du labo.

Certaines personnes furent un soutien indispensable dans mon quotidien. Sans ordre particulier, je souhaiterais remercier Nicolas et Yohann Bacquey, Alexandre Letois, Léna Coms, Axel Huet, Laureen Collette, Yohann Troude, Emily Abbott et Thibaut Vallée. Je remercie enfin toute ma famille pour son soutien inconditionnel et total.

Je remercie ensuite tous ceux que j'ai oublié de remercier : toutes ces discussions rapides dans les couloirs, ou enflammées lors de colloques, toutes ces rencontres éphémères et tous ces amis lointains. Vous avez apporté, à votre manière, une pierre à mon modeste édifice.

Enfin, merci à toi lecteur de cette thèse, de t'intéresser à mon travail de recherche. Ton regard critique lui donne un sens et une raison d'exister.

Table des matières

Chapitre 1

Introduction

En 2016, il y a plus de 1,71 milliards de comptes actifs sur un réseau social comme Facebook [1]. Le nombre d'inscrits sur les réseaux sociaux sur le web augmente de deux cent millions par an depuis 2010 [2]. L'augmentation du nombre de comptes entraine mécaniquement une augmentation du volume de données produit. La problématique de la volumétrie des données est au coeur des challenges algorithmiques liés aux réseaux sociaux.

La taille des réseaux sociaux complique leur analyse. Pour pallier ce problème, plusieurs méthodes sont envisageables, l'une d'elles est de partitionner les réseaux sociaux en éléments plus petits. Ce partitionnement permet alors, dans l'idéal, de comprendre le comportement global du système en analysant localement les éléments qui le composent et leurs comportements. Dans cette thèse, je m'intéresse au partitionnement de réseaux sociaux en sous-groupes d'individus le composant.

J'ai choisi de travailler avec une représentation usuelle des réseaux sociaux : le graphe. Un graphe est un ensemble d'entités et d'interactions (on parle traditionnellement de noeuds et d'arêtes). Dans les réseaux sociaux web, ces interactions sont représentées explicitement. Par exemple, sur Facebook on se déclare ami d'une personne tandis que sur Twitter on suit son fil d'actualité. La Fig. 1.1 illustre un extrait du réseau de Facebook [65] représenté sous forme de graphe. Les objets d'étude principaux de ma thèse sont les structures formées par ces interactions. Elles permettent de caractériser des groupes d'individus.

Seule une fraction de l'activité des individus dans les réseaux sociaux est représentée par ce modèle. En effet, dans ce modèle, on ne fait pas apparaître

1. https://investor.fb.com/investor-news/press-release-details/2016/Facebook-Reports-Second-Quarter-2016-Results/default.aspx
2. https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

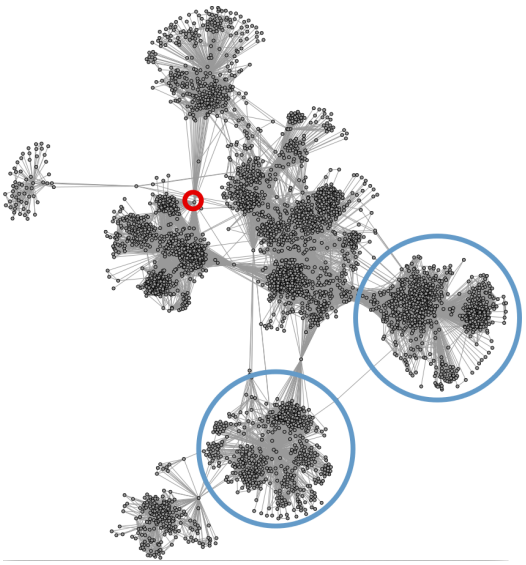


FIGURE 1.1 – Un extrait de Facebook [65] sous forme de graphe : 4 039 utilisateurs et leurs 88 234 relations d'amitié.

le contenu des messages, le temps en ligne, ou même la nature multiple des relations (travail, amis, famille, etc.). Cependant, réduire le réseau de cette manière permet d'observer et d'interpréter des caractéristiques de la structure. Voici par exemple quelques observations que l'on peut réaliser à partir de la figure Fig. 1.1 :

1. Certains individus sont à la jonction de différentes parties du réseau. C'est par exemple le cas du noeud entouré en rouge. Cette position centrale permet d'agréger l'information de plusieurs groupes distincts et donne la possibilité de contacter indirectement beaucoup d'individus [41].
2. On peut aussi remarquer que, même s'il y a plus de quatre mille individus dans le réseau, le nombre d'intermédiaires pour aller de n'importe quel individu à un autre est faible, moins d'une dizaine. Une information peut ainsi emprunter un chemin court pour aller d'un point à un autre du réseau et donc se propager rapidement [24].
3. On observe que les individus forment des groupes, comme ceux entourées en bleu. Il s'agit de parties du réseau constituées d'ensembles d'individus ayant beaucoup de contacts entre eux. Il y a souvent une raison sociale à l'existence de ces groupes dans notre modèle [39].

De nombreux domaines applicatifs sont basés sur les caractéristiques sous-jacentes associées à ces trois observations [6, 7]. Les travaux de cette thèse

portent tous sur la notion de groupe, esquissée dans la troisième observation ci-dessus.

L'objectif de la *détection de communautés* est la détection de ces groupes. Il n'y a pas de consensus sur la définition de ce qu'est une communauté. Il est cependant communément admis qu'il s'agit d'une partie dense du graphe représentant le réseau social [36].

Un grand nombre d'application réelles utilise des algorithmes de détection de communautés : recherche d'organisations criminelles [33], détection de « spam » [12], systèmes de recommandation [84], etc. D'autres disciplines scientifiques utilisent la détection de communautés pour mieux comprendre la structure des objets étudiés. C'est le cas de la sociologie [93] et de la biologie [32] par exemple.

Du fait de la variété des communautés recherchées, les définitions de la communauté sont multiples. Par exemple, une entreprise cherchant à diffuser un message publicitaire dans certaines communautés sera intéressée par les facteurs facilitant la transmission. Dans un autre contexte, la détection des communautés de « spammers » utiliserait des critères de cohésion interne, où les membres d'une communauté doivent avoir des caractéristiques proches [60].

Girvan et Newman [39] définissent une communauté comme un ensemble d'entités au sein duquel il y a plus de relations internes qu'externes. Radicchi *et al.* [81] ajoutent à cette définition la contrainte que les individus d'une communauté ont plus de voisins à l'intérieur de leur communauté qu'à l'extérieur. Ils appellent ces structures les communautés fortes.

Ces définitions sont binaires : elles n'apportent pas de gradation quant au respect des propriétés recherchées. Afin de permettre plus de souplesse, on peut concevoir des fonctions attribuant un score aux groupes. On appelle ces fonctions des *fonctions de qualité*. Elles quantifient les propriétés considérées comme importantes pour l'application. Les fonctions de qualité peuvent être vues comme une extension des définitions. Newman et Girvan ont revisité leur définition pour créer une fonction de qualité [75], la modularité.

Les fonctions de qualité sont plus flexibles que les définitions binaires. En effet, elles permettent d'obtenir une préférence entre des communautés ainsi qu'entre des partitions du graphe. De nombreux algorithmes [20, 78] créent initialement une partition triviale (une communauté par individu, par exemple) et améliorent cet état par des opérations locales qui augmentent la valeur de la fonction de qualité. Les fonctions de qualité servent aussi à l'évaluation des partitions [43]. Pour évaluer un algorithme de détection de communautés, on procédera à l'évaluation des partitions qu'il calcule.

En plus des fonctions de qualité, les chercheurs utilisent fréquemment des jeux de données extraits de la réalité pour évaluer les structures communau-

taires (par exemple [52, 79]). Ces jeux de données décrivent des structures assimilées à des communautés, par exemple des domaines de recherche dans des réseaux de chercheurs [16]. L'évaluation consiste alors à évaluer la proximité entre partitions et données.

Les problématiques au coeur de cette thèse sont les suivantes :

— La diffusion de l'information sur les réseaux sociaux est très proche de modèles infectieux. Est-il possible de s'inspirer de ces modèles pour détecter les communautés ?

— Les fonctions de qualité et les jeux de données sont utilisés pour évaluer les partitions de deux manières différentes. Quel est le lien entre les deux méthodes évaluatives ?

— Les communautés ont été assez peu caractérisées dans le cas où le réseau correspond à un ensemble de messages. Dans ces réseaux, je m'intéresse aux messages liés causalement, comme dans une conversation. La structure communautaire a-t-elle une influence sur la façon dont les messages s'enchainent ?

Les contributions de cette thèse sont les réponses à ces problématiques.

Le chapitre 2 contient les notions nécessaires à la lecture de cette thèse. J'y présente d'abord les notions fondamentales de la théorie des graphes puis les propriétés classiques des réseaux sociaux. Enfin, j'y décris l'ensemble des algorithmes de détection de communautés, fonctions de qualité et méthodes de comparaison que j'utilise dans ce manuscrit.

Le chapitre 3 présente une étude expérimentale des réseaux formés par des messages. Les expériences qui y sont menées ont pour but de déterminer le lien entre la structure communautaire et les messages liés causalement.

Le chapitre 4 introduit une nouvelle fonction de qualité : la compacité. Elle est basée sur les propriétés d'un modèle simple de diffusion de l'information. Cette fonction mesure dans ce modèle le taux de propagation d'un message dans une communauté. Une forte compacité implique que l'information se propage rapidement dans la communauté.

Le chapitre 5 contient un nouvel algorithme de détection de communautés. Celui-ci repose sur un algorithme de parcours de graphes visitant les zones denses les unes après les autres. Cet algorithme crée des « clusters » compacts.

Le chapitre 6 présente une étude sur la correspondance entre l'évaluation par les fonctions de qualité et la comparaison aux jeux de données. J'y définis la notion de contexte, c'est-à-dire d'ensemble de jeux de données qui se comportent de manière similaire au regard des fonctions de qualité.

Le chapitre 7 présente un outil développé dans le cadre de cette thèse : CoDACom[3] (« Community Detection Algorithme Comparator »). Cet outil permet d'analyser des algorithmes de détection de communautés et des fonctions de qualité. CoDACom a plusieurs objectifs : analyser de nouveaux algorithmes, tester un algorithme connu dans un cas d'étude nouveau, et évaluer les méthodes d'analyse.

Enfin, le chapitre 8 conclut la thèse et ouvre quelques perspectives.

3. `http://codacom.greyc.fr`

Chapitre 2

Rappels

Dans ce chapitre, je rappelle les notions essentielles à la lecture de cette thèse. J'introduis d'abord les concepts fondamentaux de la théorie des graphes, utilisés ici pour représenter et manipuler les réseaux sociaux. Je présente ensuite les propriétés structurelles connues des réseaux sociaux. Enfin je présente l'état de l'art des algorithmes de détection de communautés, fonctions de qualité et méthodes de comparaison.

2.1 Éléments de théorie des graphes

Dans cette section, je présente les définitions de quelques éléments de théorie des graphes. Les notations utilisées sont listées Tab. 2.1 page 13.

Définition 2.1 (Graphe)**.** *Un graphe $G = (V, E)$ est un ensemble V de noeuds et un ensemble $E \subseteq V \times V$ d'arêtes entre ces noeuds.*

Sauf mention contraire, je considère des graphes non orientées, c'est-à-dire que pour tout $u, v \in V$, (u, v) est indistinguable de (v, u). Les graphes utilisés sont sans boucle, c'est-à-dire : $\forall u \in V$, $(u, u) \notin E$. Par convention, je note $n = |V|$ le nombre de noeuds et $m = |E|$ le nombre d'arêtes.

Définition 2.2 (Incidence)**.** *Une arête $(u, v) \in E$ est dite incidente à u et à v, et ces noeuds sont dit voisins.*

Définition 2.3 (Degré)**.** *Le degré d'un noeud v est le nombre d'arêtes incidentes à ce noeud, $k_v = |\{e \mid e \in E, v \in e\}|$.*

Notons que dans la définition 2.3, le degré devrait être formellement noté $k_v(G)$. Cependant, pour simplifier la lecture, le graphe ne sera pas mentionné quand il n'y a pas d'ambiguïté. Dans le cas contraire, le graphe sera précisé en indice. Par exemple, je noterai $k_{v,G}$ le degré du noeud v dans G.

Définition 2.4 (Densité)**.** *La densité d'un graphe est la probabilité que deux noeuds pris au hasard soient voisins.*

Un exemple de graphe est présenté Fig. 2.1.

$$\begin{aligned} &\mathrm{G} = (\mathrm{V},\ \mathrm{E}) \\ &\mathrm{V} = \{\mathrm{v}_1,\mathrm{v}_2,\mathrm{v}_3,\mathrm{v}_4,\mathrm{v}_5\} \\ &\mathrm{E} = \{(\mathrm{v}_1,\mathrm{v}_2),(\mathrm{v}_2,\mathrm{v}_3),(\mathrm{v}_2,\mathrm{v}_5),(\mathrm{v}_3,\mathrm{v}_4),(\mathrm{v}_4,\mathrm{v}_5)\} \end{aligned}$$

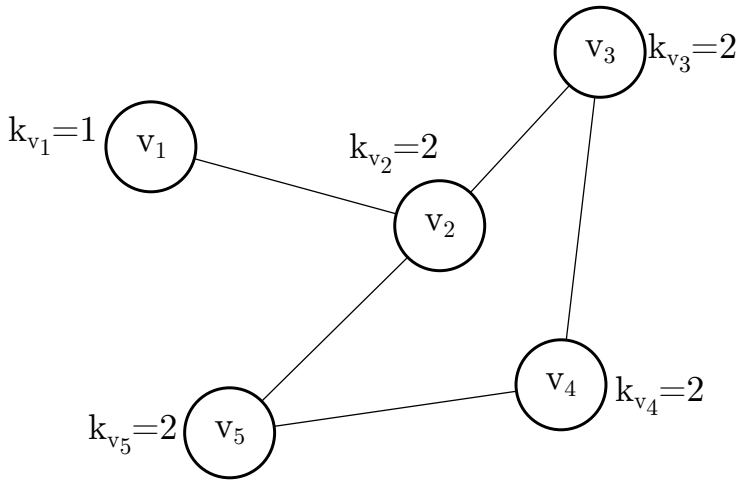


FIGURE 2.1 – Exemple de graphe, représenté de manière ensembliste et graphique. Le degré de chaque noeud est indiqué. La densité de ce graphe est de 0,5.

Définition 2.5 (Clique)**.** *Une clique est un graphe de densité 1. Une k-clique est une clique ayant k noeuds.*

J'utilise parfois des extensions du graphe, comme le graphe orienté ou le graphe pondéré.

Définition 2.6 (Graphe orienté)**.** *Un graphe orienté $G = (V, E)$ est un graphe pour lequel $\forall u,v \in V$, (u, v) est distinguable de (v, u)*

Dans un graphe orienté, on parle d'arc et non d'arête. Dans la suite de ce manuscrit, je dirai qu'un arc (u, v) est *émis* par $u \in V$ et est *reçu* par $v \in V$.

Définition 2.7 (Graphe pondéré)**.** *Un graphe pondéré $G = (V, w)$ est un ensemble de noeuds V et une fonction de poids $w : V \times V \to \mathbb{R}$ qui associe une valeur réelle à chaque couple de noeuds.*

2.1.1 Chemins

Dans cette thèse, j'utilise des algorithmes visitant des noeuds de voisins en voisins. Ces visites successives forment des *chemins*.

Définition 2.8 (Chemin). *Un chemin de longueur $k \in \mathbb{N}^+$ dans un graphe est une suite de noeuds $\pi = (v_0, \ldots, v_k) \in V^{k+1}$ respectant $\forall i \in [0..k-1], (v_i, v_{i+1}) \in E$.*

Je note $\Pi(u,v)$ l'ensemble des chemins possibles entre u et v et je note $len(\pi)$ la longueur du chemin π.

Définition 2.9 (Composante connexe). *Une composante connexe d'un graphe est un ensemble maximal (au sens du cardinal) de noeuds $V' \subseteq V$ tel que $\forall u, v \in V'$, $\Pi(u,v) \neq \emptyset$.*

Je traite principalement de graphes *connexes*, qui ne comportent qu'une composante connexe.

Définition 2.10 (Distance). *La distance entre deux noeuds dans la même composante connexe est $dist(u,v) = \min_{\pi \in \Pi(u,v)} len(\pi)$.*

Définition 2.11 (Excentricité). *L'excentricité d'un noeud dans un graphe connexe est la distance maximale entre ce noeud et les autres noeuds du graphe, $exc(v) = \max_{u \in V} dist(u,v)$.*

Définition 2.12 (Diamètre). *Le diamètre d'un graphe connexe est le maximum des distances entre les noeuds, $diam(G) = \max_{u,v \in V} dist(u,v)$.*

2.1.2 Partitions et couvertures

L'objectif de la détection de communautés est de découvrir des groupes d'individus dans un réseau social. Ces groupes sont associés à des « clusters » dans le graphe.

Définition 2.13 (Cluster). *Un « cluster » est un ensemble de noeuds non vide.*

Un « cluster » est dit *trivial* s'il ne contient qu'un noeud.

Définition 2.14 (Internalité). *Une arête $(u,v) \in E$ est dite interne à un « cluster » c si $u \in c$ et $v \in c$. Elle est dite externe si $u \in c$ ou $v \in c$.*

Définition 2.15 (Couverture). *Une couverture C est un ensemble de « clusters » recouvrants $\bigcup_{c \in C} c = V$.*

Deux « clusters » ayant un noeud en commun sont dit *chevauchants.*

Définition 2.16 (Partition)**.** *Une partition C d'un graphe est une couverture dont les « clusters » sont disjoints :* $\forall c_1, c_2 \in C,\ c_1 \neq c_2 \Rightarrow c_1 \cap c_2 = \emptyset$

Dans cette thèse, je m'intéresse aux algorithmes de détections de communauté produisant des partitions [36]. Je note toutefois qu'il existe des algorithmes de détection de communautés produisant des couvertures [102].

Définition 2.17 (Sous-graphe induit)**.** *Le sous-graphe induit par un « cluster » c, est un graphe $G' = (c, E')$ où $E' = c^2 \cap E$.*

J'étends aux « clusters » les définitions propres aux graphes, comme la connexité ou le diamètre. Ces fonctions sont alors à appliquer au sous-graphe induit par le « cluster ».

J'étudie systématiquement des « clusters » connexes.

Définition 2.18 (Graphe quotient)**.** *Le graphe quotient associé à la partition C d'un graphe $G = (V, E)$ est un graphe pondéré $G' = (C, w)$ tel que* $\forall c_1, c_2 \in C,\ w(c_1, c_2) = |\{(a, b) \mid (a, b) \in E, a \in c_1, b \in c_2\}|$.

2.1.3 Parcours et marches aléatoires

Un parcours de graphe est un chemin visitant chaque noeud du graphe. Un *algorithme de parcours de graphe* visite donc séquentiellement tous les noeuds du graphe.

Dans cette thèse, j'utilise le *DFS* (« Depth-First Search », parcours en profondeur) ainsi que le *BFS* (« Breadth-First Search », parcours en largeur).

Le DFS visite un noeud v puis choisit comme noeud à visiter par la suite l'un des voisins non visités de v. Si tous les voisins de v ont déjà étés visités, le DFS choisit l'un des voisins du noeud précédent, et ainsi de suite. Le DFS est présenté Alg. 1.

Algorithme 1 : DFS(G,s)

Données : Un graphe G et un noeud de départ s

```
1 début
2     Marque s comme visité
3     pour v ∈ voisins(s) faire
4         si v n'est pas marqué comme visité alors
5             DFS(G, v)
```

Le BFS place d'abord le noeud d'origine dans une file. À chaque itération, le BFS va visiter le premier élément de la file puis placer tous ses voisins dans la file, s'ils n'y sont pas déjà. Le BFS est présenté Alg. 2.

Algorithme 2 : BFS(G,s)

Données : Un graphe G et un noeud de départ s

```
1 début
2     file ← (s)
3     tant que file ≠ () faire
4         v ← défile(file)
5         pour u ∈ voisins(v) faire
6             si u n'est pas marqué comme enfilé alors
7                 marque u comme enfilé
8                 enfile(file, u)
```

Le BFS visite les noeuds par ordre de distance par rapport au noeud initial. Le DFS commence par créer un long chemin, puis revient sur ses pas quand il ne peut plus continuer. Un exemple de l'exécution de ces deux algorithmes est présenté Fig. 2.2

Plusieurs algorithmes de détection de communautés utilisent des marches aléatoires uniformes, un processus discret sur le graphe. À chaque instant, un marcheur est situé sur un noeud du graphe. Il se déplace à l'instant suivant uniformément aléatoirement vers l'un des voisins du noeud sur lequel il est situé. Le chemin formé est appelé une marche aléatoire. Les probabilités de transition du marcheur aléatoire sur un graphe sont illustrées Fig. 2.3.

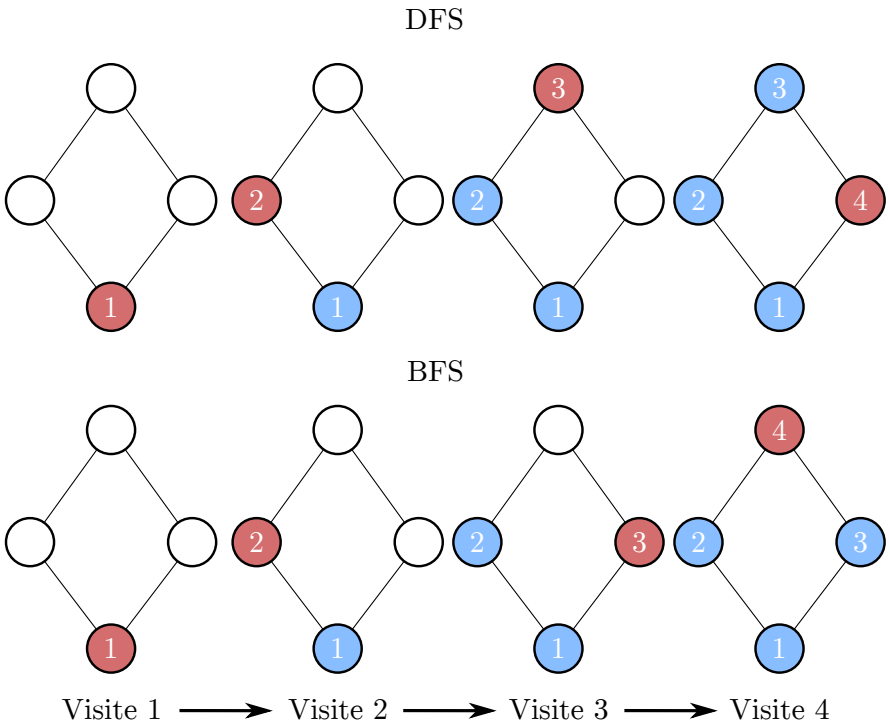


FIGURE 2.2 – État d'un graphe lors d'une exécution du DFS et du BFS. Les noeuds bleus représentent les noeuds visités et les noeuds rouges sont les derniers noeuds visités. L'itération correspondant à la visite est indiquée dans le noeud. Les états successifs du graphe sont représentés de gauche à droite.

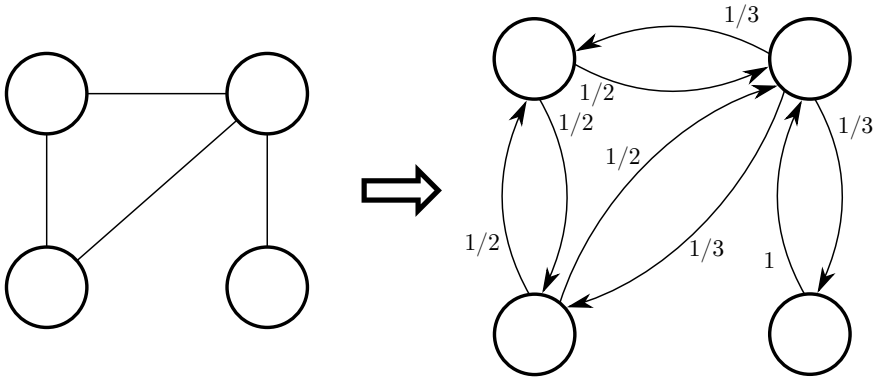


FIGURE 2.3 – Un graphe et les probabilités de transition d'un marcheur aléatoire sur ce graphe.

Notation	Définition
$G = (V,E \in \{u,v \mid u \in V, v \in V\})$	Un graphe, un ensemble V de noeuds et un ensemble E d'arêtes
$n = \lvert V \rvert$	Le nombre de noeuds
$m = \lvert E \rvert$	Le nombre d'arêtes
$k_v = \lvert \{e \mid e \in E, v \in e\} \rvert$	Degré d'un noeud v
$len(\pi) = \lvert \pi \rvert - 1$	Longueur d'un chemin π
$\Pi(u,v)$	L'ensemble des chemins allant de u à v
$dist(u,v) = \min_{\pi \in \Pi(u,v)} len(\pi)$	Distance entre deux noeuds u et v
$exc(v) = \max_{u \in V} dist(u,v)$	Excentricité du noeud v
$diam(G) = \max_{u,v \in V} dist(u,v)$	Diamètre d'un graphe G
$Vol(c) = \sum_{v \in c} k_v$	La somme des degrés d'un « cluster »
$m(c) = \lvert \{u,v \mid u,v \in c, \{u,v\} \in E\} \rvert$	Le nombre d'arêtes internes d'un « cluster »
$m(c,c') = \lvert \{i,j \mid (i,j) \in E, i \in c, j \in c'\} \rvert$	Le nombre d'arêtes entre les « clusters » c et c'
$c(C,v) = \{c \mid c \in C, v \in c\}$	Le « cluster » dans lequel apparaît le noeud v dans la partition C
$\mathcal{C}(V)$	Ensemble des partitions d'un ensemble V

TABLE 2.1 – Notations

2.2 Réseaux sociaux

Un « réseau social » est un réseau ayant été créé par des interactions sociales [31]. Il existe différentes catégories de réseaux sociaux. Dans les graphes représentant ces réseaux, les individus correspondent aux noeuds et leurs interactions correspondent aux arêtes. Je distingue les interactions bilatérales, nécessitant l'accord des deux individus pour être établie, des interactions unilatérales.

Les **réseaux sociaux web** permettent d'établir explicitement des relations entre les utilisateurs. Les relations dans certains de ces sites web sont bilatérales. C'est par exemple le cas de Facebook. Dans d'autres réseaux comme Twitter ou Youtube, la relation sociale est établie de manière unilatérale par le « follow » (le fait de suivre les publications de quelqu'un). Deux individus sont voisins dans le graphe associé si une relation existe entre les deux.

Les **réseaux de communication** sont formés par des transmissions d'information entre individus. On retrouve la même distinction entre unilatéralité (comme pour les e-mails et les SMS) et bilatéralité (appels téléphoniques et vidéoconférences). Deux individus sont voisins dans le graphe associé s'ils ont communiqué en utilisant ce réseau.

Les **réseaux de collaboration** correspondent à des individus ayant travaillé ensemble sur un sujet. On y trouve par exemple des réseaux d'acteurs ayant tourné des films ensemble ou des scientifiques ayant coécrit des articles. Les relations dans ce type de réseau sont bilatérales. Deux individus sont voisins dans le graphe associé s'ils ont collaboré.

On a observé des caractéristiques communes aux graphes représentant les réseaux sociaux.

Tout d'abord, ces graphes sont d'une densité faible. Les modèles classiques supposent même un degré moyen constant [98, 8]. Leskovec *et al.* [62] nuancent cette supposition en observent sur des réseaux évoluant au court du temps que $m \in \mathcal{O}(n^a)$, avec a mesuré entre 1,05 et 1,68.

Ensuite, on observe que les distances dans ces graphes sont courtes [61]. Ainsi, en passant de voisin en voisin, il est possible d'atteindre n'importe quel autre point du graphe en un nombre faible d'intermédiaires, même si le graphe en question a beaucoup de noeuds. Ils s'agit de l'une des caractéristiques des graphes petit monde [70].

La distribution du degré des noeuds est proche d'une loi de puissance [8]. La probabilité qu'un noeud aléatoire du graphe soit de degré k est proche de $k^{-\gamma}$, avec γ dépendant du graphe, le plus souvent entre 2 et 4. Cette loi de

probabilité a pour conséquence qu'il existe une grande diversité des degrés des noeuds, avec quelques noeuds à degré très important et un grand nombre de noeuds à degré faible.

Une propriété sociologique importante dans les réseaux sociaux est l'homophilie [68] : les individus connaissent des personnes qui leur sont similaires. La conséquence structurelle est que l'on observe beaucoup de triangles, c'est-à-dire des triplets d'individus formant des 3-cliques. En effet, si deux individus a et b se connaissent, et que c connaît a mais pas de b, c a la volonté et l'occasion de former un lien avec b :

— Si a et b sont similaires, et que a et c sont similaires, il est probable que c soit similaire à b.

— Du fait de la relation de c avec a, c a l'opportunité de former un lien avec b.

Cette propriété explique aussi l'apparition de sous-graphes denses dans les réseaux sociaux.

2.3 Fonctions de qualité

Dans cette section, je décris les fonctions de qualité que j'utilise pour étudier les communautés.

Yang et Leskovec [103] ont défini quatre caractéristiques qu'ils estiment axiomatiquement « désirables » dans les communautés recherchées.

— La densité interne : les noeuds à l'intérieur de la communauté sont très connectés entre eux.

— La séparabilité : la communauté présente des caractéristiques différentes de son entourage. Par exemple, les noeuds de la communauté ont plus de voisins à l'intérieur qu'à l'extérieur de celle-ci.

— La cohésion interne : les caractéristiques de la communauté sont robustes à la suppression de noeuds ou d'arêtes. Par exemple, il est nécessaire de supprimer un grand nombre des arêtes d'une communauté pour qu'elle ne soit plus connexe.

— La fermeture triadique : pour u, v, w des noeuds de la communauté, si $(u, v) \in E$ et $(v, w) \in E$, alors généralement $(u, w) \in E$.

Ces caractéristiques ne sont pas complètement indépendantes, par exemple une forte densité interne est souvent corrélée avec une fermeture triadique importante.

Une fonction de qualité est une application $q(\mathcal{C}(V)) \to \mathbb{R}$ qui quantifie ces caractéristiques sur une partition afin d'obtenir un résultat numérique.

J'utilise la convention que les fonctions de qualité doivent retourner un haut score pour des « clusters » ayant des caractéristiques considérées comme préférables par les auteurs de ces fonctions. Je normalise les quelques fonctions de qualité ayant un comportement inverse. Comme ces fonctions de qualité sont systématiquement comprises entre 0 et 1, pour une fonction de qualité non normalisée q_f j'utilise la fonction $q_{1-f} = 1 - q_f$.

Les fonctions de qualité sont présentées Tab. 2.2 page 18.

La **Surprise** [1] et la **Signifiance** [92] sont basées sur le calcul d'une dissimilarité de distribution, la divergence de Kullback-Leibler [57]. Dans notre cas, elle n'est utilisée que pour des distributions binaires (un évènement et son complément). Il s'agit de la quantité d'information perdue quand la distribution de référence est utilisée pour approcher l'autre distribution. Je note y comme la probabilité de l'un de ces deux évènements dans la distribution de référence et x la même probabilité dans l'autre distribution. La divergence de Kullback-Leiber est décrite Eq. 2.1.

$$D(x||y) = x \log(\frac{x}{y}) + (1-x) \log(\frac{1-x}{1-y}) \tag{2.1}$$

La distribution de référence de la Surprise correspond à l'évènement qu'une paire de noeuds aléatoires soit à l'intérieur de la même communauté. Elle est comparée à la distribution de l'évènement où une arête aléatoire est interne à une communauté. La Signifiance est une somme de divergences, une par communauté, dont la distribution de référence est toujours la même : celle de l'évènement qu'une paire de noeuds aléatoires soient reliés dans le graphe. L'autre distribution de chaque divergence correspond à la probabilité qu'une paire de noeuds soit connectée dans la communauté considérée. Ces deux fonctions mesurent la densité interne. J'utilise la définition de [91] qui approxime asymptotiquement la valeur de la surprise.

Certaines fonctions de qualité sont définies au **niveau de la communauté**, c'est-à-dire qu'elles calculent une qualité pour chaque « cluster ». Leur entrée est donc étendue pour inclure un « cluster » c en plus de la partition C. Je prends la somme de cette fonction sur tous les « clusters » pour la calculer sur la partition entière.

Le **Cut-ratio** [99] correspond à la probabilité qu'il existe une arête $(u, v) \in E$ pour deux noeuds aléatoires u et v tels que $u \in c$ et $v \in V \backslash c$. Cette fonction mesure la séparabilité.

La **Conductance** [51] correspond à la probabilité qu'une arête aléatoire $(u, v) \in E$ telle que $|\{u, v\} \cap c| \geq 1$ soit externe. Cette fonction mesure la séparabilité.

Je fais le choix de pondérer ici les mesures de conductance et de cut-ratio avec la taille de la communauté en nombre de noeuds. En effet, ces fonctions sont pensées originellement pour être appliquées au niveau de la communauté, et pas pour être agrégées au niveau de la partition. Cette pondération a pour but qu'une communauté ait une influence proportionnelle à sa taille sur le score aggloméré.

La **Modularité** a été proposée par Girvan et Newman [39]. Comme je le détaille Sec. 2.4, plusieurs algorithmes très utilisés sont basés sur cette fonction [9, 20]. Elle mesure la différence entre le nombre d'arêtes internes à la communauté (le premier terme) et l'espérance de cette valeur dans le modèle de configuration (le second terme). Le modèle de configuration (ou « configuration model ») correspond au même graphe dont les arêtes sont coupées en deux et ces demi-arêtes sont rattachées de manière aléatoire. Ainsi, si un groupe d'individus a une densité de connexion significativement supérieure à ce que le modèle prévoit, ces connexions ne sont pas explicables par des branchements aléatoires.

La modularité quantifie la densité interne ainsi que la séparabilité. La densité interne est représentée par le nombre d'arêtes internes. On remarque aussi qu'une augmentation du nombre d'arêtes externes entraîne une diminution de la modularité, du fait de la comparaison avec le modèle. La modularité mesure donc aussi la séparabilité.

La **Modularité locale** [73] est similaire à la modularité, mais le modèle de configuration est modifié. Les demi-arêtes de ce modèle ne peuvent être rattachées qu'entre des communautés voisines, notées $voisins(c)$. La modularité locale mesure la densité interne ainsi que la séparabilité.

La **Densité de modularité** [19] consiste en la modularité dont on lui soustrait une pénalité suivant sa connectivité externe. Le but est de mettre l'accent sur la propriété de séparabilité, tout en conservant la mesure de la densité interne.

Certaines fonctions de qualité sont définies au **niveau du noeud**, c'est-à-dire qu'elles calculent une qualité pour tous les noeuds du graphe. Leur entrée est donc étendue pour inclure un noeud v en plus du « cluster » c dans lequel il est et de la partition C. Afin de calculer la fonction sur toute la partition, la moyenne de la fonction de qualité est prise.

Le **Coefficient local interne de clustering** [98] (appelé coefficient de clustering dans ce manuscrit) d'un noeud est la probabilité que deux de ses voisins pris aléatoirement dans la même communauté soient aussi voisins entre eux. Cette fonction mesure la fermeture triadique.

Cette fonction est utilisée dans la **Permanence** [18], où elle est addition-

Nom	Fonction
Coefficient de clustering	$q_{clus}(C,c,v) = \frac{2\lvert\{u,v,w \in c, ((v,u),(v,w),(u,v)) \in E^3\}\rvert}{\lvert c\rvert(\lvert c\rvert - 1)}$
Permanence	$q_{perm}(C,c,v) = \frac{m(v,c)}{\max_{c' \in C \setminus \{c\}}(m(v,c')) \times k_v} + q_{clus}(C,c,v) - 1$
1-Flake-ODF	$q_{1-flak}(C,c,v) = \begin{cases} 1 & \text{quand } m(v,C) > m(v, V \setminus c) \\ 0 & \text{sinon} \end{cases}$
FOMD	$q_{FOMD}(C,c,v) = \begin{cases} 1 & \text{quand } m(v,c) > d_m \\ 0 & \text{sinon} \end{cases}$
1-Cut ratio	$q_{1-cut}(C,c) = \left(1 - \frac{m(c, V \setminus c)}{\lvert c\rvert(n - \lvert c\rvert)}\right) \times \frac{\lvert c\rvert}{n}$
1-Conductance	$q_{1-cond}(C,c) = \left(1 - \frac{m(c, V \setminus c)}{Vol(c)}\right) \times \frac{\lvert c\rvert}{n}$
Modularité	$q_{mod}(C,c) = \frac{m(c)}{m} - \left(\frac{Vol(c)}{2m}\right)^2$
Modularité locale	$q_{lmod}(C,c) = \frac{m(c)}{m(c \cup voisins(c))} - \left(\frac{Vol(c)}{2 \cdot m(c \cup voisins(c))}\right)^2$
Densité de modularité	$q_{dmod}(C,c) = q_{mod}(C,c) - \sum_{c' \in C \setminus c} \frac{m(c,c')}{m}$
Surprise	$q_{surp}(C) = D\left(\frac{\sum_{c \in C} m(c)}{m} \middle\| \frac{\sum_{c \in C} \binom{\lvert c\rvert}{2}}{\binom{n}{2}}\right)$
Signifiance	$q_{sign}(C) = \sum_{c \in C} \binom{\lvert c\rvert}{2} D\left(\frac{m(c)}{\binom{\lvert c\rvert}{2}} \middle\| \frac{m}{\binom{n}{2}}\right)$

TABLE 2.2 – Fonctions de qualité

née à une mesure de la connectivité du noeud dans sa communauté. Cette densité est pondérée de telle manière qu'un noeud a une permanence plus faible s'il est relativement très connecté à une autre communauté en particulier. La Permanence mesure la séparabilité ainsi que la fermeture triadique.

La **Flake-ODF** [34] (« Out-Degree Fraction ») compare directement degré interne et degré externe. Cette fonction mesure la séparabilité.

La **FOMD** [103] (« Fraction Over Median Degree ») compare le degré interne du noeud avec le degré médian d_m du graphe. Cette fonction mesure la densité interne.

2.4 Algorithmes de détection de communauté

Dans cette section, je présente les algorithmes de détection de communautés auxquels je fait référence au long de cette thèse.

2.4.1 Stratégies de partitionnement

Dans cette partie, je présente les différentes stratégies adoptées par les algorithmes de détection de communautés.

Les méthodes de **partitionnement de données** s'inspirent directement du problème éponyme, où l'on cherche à grouper des objets génériques. La stratégie dans ce domaine est typiquement d'établir une fonction de similarité entre les objets, puis d'exécuter un algorithme chargé de trouver des « clusters » où les similarités internes sont fortes [50]. On y trouve des techniques de partitionnement hiérarchique [20] (les communautés fusionnent ou se divisent au fur et à mesure de l'exécution) et de coupure de graphe [35] (en minimisant le nombre d'arêtes entre les parties). On y trouve aussi l'intégration des graphes dans un espace métrique par une mesure de dissimilarité entre les noeuds [83].

Les méthodes **divisives** sont des méthodes de partitionnement hiérarchique qui se basent sur l'identification et la suppression successive des arêtes que l'on suppose être entre les « clusters ». Ces suppressions déconnectent le graphe et les composantes connexes sont les communautés.

Newman et Girvan ont introduit la **modularité** [75] qui inspira une série de méthodes basées sur celle-ci. Il a été prouvé par Brandes *et al.* [10] que trouver la partition optimale en modularité est un problème NP-Complet, ce qui rend le passage à l'échelle extrêmement coûteux. Des approches d'optimisation gloutonne ont étés développées pour apporter une alternative acceptable en temps de calcul, comme l'algorithme de Louvain [9]. D'autres approches existent, comme la méthode du recuit simulé [42] ou l'analyse spectrale [74].

En effet, de nombreux algorithmes sont basés sur des méthodes **spectrales**. Le spectre d'une matrice spécifique qui définit une notion de proximité entre les noeuds y est analysé. Les vecteurs propres associés aux valeurs propres les plus faibles (sauf le premier, à valeur propre nulle) décrivent des « clusters » à forte similarité interne [72]. En effet, prendre les k premiers vecteurs propres (en ignorant le premier) permet une projection des n noeuds dans un espace k-dimensionnel. Il suffit alors suffit d'exécuter un partitionnement classique (type k-means) pour trouver des « clusters ». La matrice Laplacienne est traditionnellement utilisée comme matrice de similarité entre les noeuds. Cependant, d'autres matrices peuvent être utilisées,

comme la matrice de modularité où la proximité est le gain en modularité apporté par la réunion des noeuds voisins dans le même « cluster » [74].

Les méthodes dites **dynamiques** simulent un processus se déroulant sur le graphe. Le modèle de Potts [101] décrit un ensemble de particules ayant chacune un état, et une notion de proximité avec les autres particules. Les particules s'influencent entre elles, celles qui sont proches les unes des autres ont tendance à partager le même état. La stratégie est alors de trouver les paramètres du modèle correspondant le mieux aux données. Un autre processus est celui de la synchronisation, où le système simulé unifie progressivement tous ses éléments au même état. Les noeuds d'une communauté étant proches les uns des autres, ils ont tendance à être localement synchronisés en premier, ce qui permet leur identification [4]. D'un autre côté, les marches aléatoires se concentrent dans les sous-graphes denses, ce qui est exploité par plusieurs algorithmes [94, 78].

Les méthodes d'**inférence statistique** font l'hypothèse que le graphe a été généré suivant un modèle, et que ce modèle admet l'appartenance des noeuds aux différentes communautés en tant que paramètre. Il s'agit alors de trouver les paramètres du modèle qui auraient généré les données observées avec la probabilité la plus importante. Par exemple, considérons que les noeuds à l'intérieur des communautés sont connectés avec une probabilité p_{in}, et à l'extérieur avec une probabilité p_{out} [45]. L'objectif est alors de trouver la partition qui aurait le plus de chances de générer le graphe observé à partir de ce modèle.

Des explications plus détaillées sur ces catégories peuvent être trouvées dans l'article récapitulatif de Fortunato [36].

2.4.2 Algorithmes

Dans cette partie, je présente les algorithmes utilisés dans ce manuscrit.

L'algorithme de **Louvain** [9] optimise la modularité de manière gloutonne. Cet algorithme déplace les noeuds dans le « cluster » de leur voisin de manière à maximiser le gain en modularité. Une fois qu'aucun gain en modularité n'est possible de cette manière, il s'applique récursivement au graphe quotient de la partition. L'algorithme se termine quand le processus récursif cesse d'améliorer la modularité.

L'algorithme de **Clauset** [20], aussi appelé Clauset-Newman-Moore (CNM) du nom de ses créateurs, est une autre approche à l'optimisation gloutonne de la modularité. Cet algorithme suit une stratégie de partitionnement hiérarchique où, à chaque étape, la fusion fournissant la plus forte augmentation de la modularité est choisie. L'algorithme s'arrête quand la partition inclut une unique communauté, et renvoie la partition maximale de modularité trouvée.

L'algorithme **MCL** [94](« Markov CLustering ») est une approche basée sur du calcul matriciel. Chaque noeud du graphe est associé à un entier différent entre 1 et n. Le graphe est représenté par une matrice A de taille $n \times n$ telle que $A_{i,j} = 1/k_j$ si $(i,j) \in E$ et $A_{i,j} = 0$ sinon. La matrice A représente un graphe pondéré dont les arêtes sont pondérées par l'inverse du degré du noeud source de l'arête.

Cette matrice est élevée au carré, ce qui a l'effet de propager les chemins : il s'agit de l'étape d'expansion. Ensuite, les éléments de la matrice sont élevés à une puissance (dépendante d'un paramètre de l'algorithme), et les colonnes sont normalisées tels que la somme de leurs éléments fasse 1. Cette étape augmente la variance du poids des arêtes : il s'agit de l'étape d'inflation. En répétant les étapes d'expansion et d'inflation, les arêtes internes des groupes d'individus fortement connectées voient leur poids augmenter tandis que le poids des arêtes externes tend vers zéro. Malgré son utilisation de la multiplication matricielle, cet algorithme a une complexité proche du linéaire sur des graphes peu denses grâce à des techniques dites de « pruning », c'est-à-dire de suppression des faibles valeurs éloignées de la diagonale.

L'algorithme **infomap** [86] associe à chaque noeud un code en deux parties : un préfixe correspondant à la communauté ainsi qu'un identifiant du noeud dans la communauté. Un chemin peut être décrit par les identifiants des noeuds qu'il parcourt, et le préfixe de communautés traversées. L'objectif est de trouver les préfixes communautaires permettant de minimiser le nombre moyens de bits nécessaires pour représenter une marche aléatoire. Cette optimisation est faite de manière gloutonne, puis raffinée par le biais d'un algorithme de recuit simulé.

Un **k-core** [87] est un ensemble maximal de noeuds liés aux autres noeuds de cet ensemble par au moins k arêtes. Les ensembles connexes du « k-core » sont considérés comme les communautés. Les « k-core » sont obtenus en supprimant successivement les noeuds de degré inférieur à k.

L'algorithme de **label propagation** [82] est un processus dynamique discret simulé sur le graphe. Il attribue initialement une étiquette contenant un identifiant différent pour chaque noeud. L'étiquette de chaque noeud est changée à chaque pas de temps par celle qui apparaît en majorité parmi ses voisins. Quand le processus cesse de modifier les étiquettes, celles-ci sont retournées comme étant les communautés.

L'algorithme de **betweenness** [39] est une méthode divisive. Le critère de suppression d'arête est la « betweenness centrality », une mesure de centralité qui compte le nombre de plus courts chemins passant par une arête. La supposition est que les plus courts chemins traversent fréquemment des arêtes entre les communautés, du fait de la différence de densité interne/externe. La suppression de ces arêtes isole donc les communautés les unes des autres.

L'algorithme **Walktrap** [78] utilise les propriétés des marches aléatoires pour établir une notion de distance entre les noeuds. Il calcule pour chaque paire de sommets (i,j) un poids $w_t(i,j)$ correspondant à la probabilité qu'une marche aléatoire aille de i à j en t étapes. Il en déduit une distance sur les paires de noeuds, décrite Eq. 2.2.

$$dist_t(i,j) = \sqrt{\sum_{v \in V} \frac{|w_t(v,i) - w_t(v,j)|^2}{k_v}} \tag{2.2}$$

Cet algorithme suit ensuite une stratégie de partitionnement hiérarchique en fusionnant les communautés à faible distance.

L'algorithme de **Spinglass** [85] modélise le graphe avec le modèle de Potts [101], décrit précédemment. Les auteurs en déduisent une fonction de score proche de celle de la modularité, qu'ils optimisent avec la méthode du recuit simulé.

L'algorithme **Conclude** [30] commence par établir une notion de distance similaire à celle Eq. 2.2. Cependant, $w_t(i,j)$ est remplacé par la probabilité que l'arc (i,j) soit traversé par une marche aléatoire de longueur t depuis une source aléatoire. Ils appliquent ensuite l'algorithme de Louvain sur le graphe pondéré qui en résulte.

L'algorithme **SCD** [79] (« Scalable Community Detection ») optimise de manière gloutonne un score mesurant la proportion des triangles présents à l'intérieur des communautés. Afin de ne pas avoir à calculer le nombre de triangles des graphes, cet algorithme approche le score en question. Comme pour l'algorithme de Louvain, cet algorithme change localement les noeuds de communautés et permet en plus aux noeuds de s'isoler dans de nouvelles communautés. Dès que le score du graphe se stabilise, l'algorithme se termine.

2.5 Méthodes de comparaison

Une méthode de comparaison (ou « extrinsic clustering evaluation metric » [3]) est une fonction évaluant la proximité entre deux couvertures du même ensemble.

La **NMI** (« Normalized Mutual Information ») est une méthode de comparaison basée sur la théorie de l'information. J'utilise la version introduite par Lancichinetti *et al.* [58].

Soient deux couvertures C et L de V. Soit X_c une variable aléatoire associée au « cluster » $c \subseteq V$ valant 1 si l'évènement qu'un noeud pris uniformément aléatoirement dans V est inclus dans c, et 0 sinon. Je note $h(x) = -x \cdot \log(x)$. L'entropie jointe $H(X_c, X_l)$ d'une paire de « clusters »

$c \in C$ et $l \in L$ est calculée ainsi :

$$\begin{aligned} H(X_c \in C, X_l \in L) &= \sum_{x \in X_c} \sum_{y \in X_l} h(p(x,y)) \\ p(X_c = 1, X_l = 1) &= \frac{|X_c \cap X_l|}{n} \\ p(X_c = 1, X_l = 0) &= \frac{|X_c| - |X_c \cap X_l|}{n} \\ p(X_c = 0, X_l = 1) &= \frac{|X_l| - |X_c \cap X_l|}{n} \\ p(X_c = 0, X_l = 0) &= \frac{n - |X_c \cup X_l|}{n} \end{aligned} \tag{2.3}$$

L'entropie conditionnelle $H(X_c|X_l) = H(X_c, X_l) - H(X_l)$ mesure la quantité d'information nécessaire pour décrire le résultat de X_c en sachant X_l. On cherche à utiliser l'entropie conditionnelle pour évaluer la proximité de deux « clusters ». Si c et l sont très similaires, cette valeur est faible. Cependant, c'est aussi le cas si l est le complément de c dans V.

En observant les termes de la somme de l'entropie jointe Eq. 2.3, les auteurs définissent la condition suivante pour considérer deux « clusters » comme pouvant être proches (je note $p(a,b) = p(X_c = a, X_l = b)$) :

$$h(p(1,1)) + h(p(0,0)) > h(p(1,0)) + h(p(1,0)) \tag{2.4}$$

En effet, ils ont remarqué que les termes $h(p(1,0))$ et $h(p(0,1))$ sont dominants quand l est le complément de c. Les deux autres termes sont dominants quand c et l sont similaires. Je note $obs(c, L)$ l'ensemble des « clusters » $l \in L$ tels que c et l satisfont la condition 2.4.

Les auteurs définissent ensuite l'entropie conditionnelle normalisée de l'ensemble $\boldsymbol{C}$ des variables aléatoires associées aux clusters de C par rapport à celles de L, notées $\boldsymbol{L}$.

$$H(\boldsymbol{C}|\boldsymbol{L})_{norm} = \frac{1}{|C|} \sum_{c \in C} \frac{\min_{l \in obs(c,L)} H(X_c|X_l)}{H(X_c)} \tag{2.5}$$

L'information mutuelle normalisée est alors définie ainsi :

$$NMI(C, L) = 1 - \frac{1}{2}[H(\boldsymbol{C}|\boldsymbol{L})_{norm} + H(\boldsymbol{L}|\boldsymbol{C})_{norm}] \tag{2.6}$$

La **F-BCubed (fb3)** [5] a deux composantes : précision et rappel. Je définis un *associé* d'un noeud v pour une couverture C comme un noeud qui a au moins un « cluster » en commun avec v dans C. La précision $prec(C, L)$

de deux partitions C et L mesure pour chaque noeud v la proportion de ses associés dans C qui sont aussi ses associés dans L. La moyenne est prise sur tous les individus. Le rappel correspond à la précision dont les partitions sont échangées, $rappel(C, L) = prec(L, C)$. Précision et rappel sont agrégés en prenant la moyenne harmonique :

$$F\text{-}BCubed(C, L) = \frac{1}{\frac{1}{2 * prec(C,L)} + \frac{1}{2 * rappel(C,L)}} \tag{2.7}$$

Amigó *et al.* [3] ont étendu cette métrique pour les couvertures, en prenant en compte le nombre de « clusters » en commun que v et ses associés ont dans les deux partitions. Je note $C_v = \{c \mid c \in C, v \in C\}$ (resp. L_v) l'ensemble des « clusters » dans C (resp. L) dans lesquels apparaît v. Je note $asso(C, v) = \{u \mid u \in V, C_v \cap C_u \neq \emptyset\}$ l'ensemble des associés de v dans C. Les auteurs définissent ainsi la précision recouvrante :

$$prec(C, L) = \frac{1}{n} \sum_{v \in V} \left[\frac{1}{|asso(C, v)|} \sum_{u \in asso(C,v)} \left(\frac{min(|C_v \cap C_u|, |L_v \cap L_u|)}{|C_v \cap C_u|} \right) \right] \tag{2.8}$$

Le rappel et la F-BCubed recouvrantes sont définis par rapport à cette précision comme pour la version originelle.

L'**Omega Index** [21] dénombre les paires de noeuds qui sont dans le même nombre de « clusters » dans les deux couvertures. Cette valeur est comparée avec son espérance dans un modèle où les deux couvertures sont formées de noeuds aléatoires. Soit $t_j(C)$ l'ensemble des noeuds ayant j « clusters » en commun dans C. L'Omega Index non ajusté est :

$$\omega_u(C, L) = \frac{1}{\binom{n}{2}} \sum_{j=0}^{\max(|C|,|L|)} |t_j(C) \cap t_j(L)| \tag{2.9}$$

La même valeur dans le modèle nul est :

$$\omega_e(C, L) = \frac{1}{\binom{n}{2}^2} \sum_{j=0}^{\max(|C|,|L|)} |t_j(C)| \cdot |t_j(L)| \tag{2.10}$$

L'Oméga Index correspond à :

$$\omega(C, L) = \frac{\omega_u(C,L) - \omega_e(C, L)}{1 - \omega_e(C, L)} \tag{2.11}$$

Chapitre 3

Étude de la structure communautaire de réseaux de communication par les motifs temporels

Dans ce chapitre, je propose une adaptation des motifs prenant en compte le délai de réception d'un message. Je déduis d'une comparaison avec un modèle de référence que les messages dans ces motifs ont des liens causaux. J'observe que ces motifs ont des propriétés différentes à l'intérieur et à l'extérieur des communautés.

Ces résultats présentés dans ce chapitre sont les premiers résultats d'un travail en cours, commun avec Remy Cazabet.

L'appartenance aux communautés n'est pas quelque chose de fixé dans le temps. Par exemple, un scientifique peut changer de domaine de recherche au cours de sa carrière. Analyser l'évolution d'un réseau au cours du temps est essentiel pour pouvoir capturer ces changements.

On définit un réseau temporel comme un réseau contenant des interactions inscrites dans le temps. Des exemples classiques de réseaux temporels sont les réseaux d'e-mails [55], les appels téléphoniques [11], la proximité physique entre des personnes équipées de puces RFID [13] et les citations dans les publications scientifiques [17]. Le fait que chaque interaction soit inscrite dans le temps permet d'observer des dynamiques qui n'apparaissent pas dans les réseaux statiques. Par exemple, il a été observé dans les réseaux d'appels téléphoniques que les utilisateurs sont très actifs durant de courtes périodes et sont inactifs le reste du temps [54].

Pour décrire ces communications, le graphe temporel peut être utilisé [48].

Définition 3.1 (Graphe temporel)**.** *Un graphe temporel $G = (V, E)$ est un ensemble de noeuds V et d'arcs $E = \{(u,v,t_1,t_2) \mid u,v \in V, t_1 \in \mathbb{R}^+, t_2 \in \mathbb{R}_{\geq t_1}\}$, où $u, v \in V$ sont la source et la destination du lien, t_1, t_2 sont respectivement la date de début et de fin de l'interaction.*

Il s'agit donc d'un graphe orienté où plusieurs arcs peuvent exister entre la même paire de noeuds. À chaque arc est associé un intervalle temporel. Je qualifie un arc $(u, v, t_1, t_2) \in E$ de *présent* à un instant t si $t_1 \leq t \leq t_2$.

Dans ce chapitre, je m'intéresse particulièrement aux réseaux formés par des communications écrites instantanées, du type messagerie en ligne, e-mails ou SMS. J'utilise donc une sous-catégorie des graphes temporels adaptée à ce type de réseau, les **flux de liens** [96]. Il s'agit d'un graphe temporel où $t_1 = t_2$.

Définition 3.2 (Flux de liens)**.** *Un flux de liens $G = (V,E)$ est un ensemble de noeuds V et d'arcs $E \subset V^2 \times \mathbb{R}^+$.*

Des algorithmes de détection de communautés ont étés conçus pour traiter les graphes temporels [66, 14]. Ils supposent qu'à chaque instant les arcs présents forment un graphe d'une densité comparable à celle d'un graphe de réseau social. Ils sont par exemple adaptés à traiter les réseaux sociaux web représentés par un graphe temporel. Les arcs de ces graphes représentent les relations, de la date de sa création à la date de sa fin. Ainsi, les arcs présents à chaque instant de ce graphe représentent des instantanés du réseau social web.

Les réseaux de communications écrites instantanées, représentés en flux de liens, ont une densité très faible presque tout le temps. En effet, le nombre de messages envoyés par des utilisateurs à chaque instant est extrêmement faible en moyenne. La stratégie actuelle de détection de communautés sur ces flux de lien consiste à agréger les arcs sur des fenêtres temporelles pour ensuite les traiter comme des graphes temporels denses [47]. Cette approche néglige une donnée des flux de liens : l'ordre des communications à l'intérieur des fenêtres d'agrégation. Or, s'il existe une relation entre l'ordre des communications et l'appartenance communautaire des noeuds impliqués dans ces communications, cela implique que les algorithmes actuels négligent une donnée importante.

Dans le but d'observer cette relation, je m'intéresse aux motifs temporels, définis par Zhao *et al.* [106]. Ces motifs correspondent à la structure formée

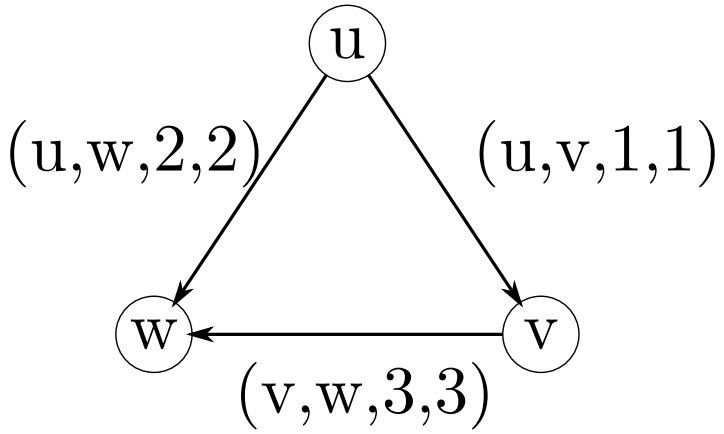


FIGURE 3.1 – Exemple d'une série de communications respectant le motif "AB-AC-BC"

par l'enchainement des arcs sur une courte période. Une instance d'un motif consiste en un ensemble d'arcs dans le graphe temporel. Soit $(u, v, t_1, t_2) \in E$ l'un des arcs d'une instance d'un motif. Les arcs ayant étés émis dans un intervalle de temps W par u ou par v peuvent aussi être inclus dans cette instance.

Définition 3.3 (Instance de motif)**.** *Une instance d'un motif sur une fenêtre de taille* $W \in \mathbb{R}^+$ *est un graphe temporel* $G = (V, E)$ *tel que* $\forall (u_i, v_i, t_{1i}, t_{2i}) \in E$, $\exists (u_j, v_j, t_{1j}, t_{2j}) \in E$ *tel que* $(u_i, v_i, t_{1i}, t_{2i}) \neq (u_j, v_j, t_{1j}, t_{2j})$, $\{u_i, v_i\} \cap \{u_j, v_j\} \neq \emptyset$ *et* $0 < t_{1i} - t_{2j} < W$ *ou* $0 < t_{1j} - t_{2i} < W$.

Un motif est une classe d'équivalence de ces instances dans le graphe temporel étudié. Deux instances de motifs $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ sont considérés comme équivalentes s'il existe une application bijective $f : V_1 \to V_2$ telle que :

$$\forall u, v \in V_1, \left|\{(u, v, t_a, t_b) \in E_1\}\right| = \left|\{(f(u), f(v), t_a, t_b) \in E_2\}\right|$$

Cette fonction d'équivalence a été étendue par Kovanen *et al.* [56] pour prendre en compte l'ordre des arcs à l'intérieur du motif.

Je fais référence aux différents motifs par les noeuds qui composent la séquence temporelle des arcs. Les noeuds sont identifiés par les lettres de l'alphabet dans l'ordre de leur apparition dans le motif. Par exemple, l'ensemble de trois arcs $\{(u,v,1,1), (u,w,2,2), (v,w,3,3)\}$ est une instance du motif "AB-AC-BC" avec la fenêtre adaptée, comme illustré Fig. 3.1.

L'apparition d'instances de motifs correspond à un type de communication qui se déroule entre les utilisateurs représentés par les noeuds. Par exemple, une conversation entre deux individus induit un motif de va-et-vient [106] (type "AB-BA-AB").

Les études au sujet des motifs temporels [106, 89, 104] mettent en avant la sur-représentation de certains motifs par rapport aux autres, comme la chaine (type "AB-BC-CD"), le va-et-vient ou l'étoile (type "AB-AC-AD"). Néanmoins, ces observations peuvent très bien provenir de propriétés qui n'ont aucun rapport avec la causalité des communications, c'est par exemple le cas de la chaine. On peut donc se poser la question de l'influence de la causalité sur les motifs.

Dans ce but, j'utilise un modèle où la causalité n'existe pas entre les messages émis. Ce modèle, que j'appelle modèle nul, sert de base de comparaison pour quantifier l'influence de la causalité. Le modèle nul est un modèle de génération de graphe. Il prend en entrée un graphe et fait disparaître une partie de ses propriétés [56] (« null model »). Dans le cadre de mes travaux, j'utilise le « correlation-mixing model » [89] pour générer des graphes où les liens causaux entre les messages sont supprimés. Ce modèle rend aléatoire l'ordre des messages émis par les noeuds.

Notons que le modèle nul est un modèle de génération aléatoire de graphe. Il est théoriquement possible de générer le graphe de base à partir du modèle. Néanmoins, la probabilité d'obtenir le graphe de base est très faible.

Dans le « correlation-mixing model », les propriétés structurelles telles que le nombre d'arcs entre deux noeuds ou leur degré sont conservées. Certaines propriétés temporelles persistent aussi, comme la fréquence d'émission ou la distribution temporelle des messages. D'un autre côté, l'enchaînement des messages est dû au hasard et non plus à un lien causal.

Pour chaque mesure sur les motifs, je compare la valeur sur le graphe d'origine et sur les graphes générés par le modèle nul. Je considère les différences statistiquement significatives entre le modèle nul et la réalité observée comme une conséquence de la causalité, comme décrit par Tabourier *et al.* [89]. Dans ce but, j'étudie la distribution de probabilité sur le modèle nul des différentes mesures en générant plusieurs instances du modèle.

J'observe en pratique que ces mesures sont distribuées suivant une loi normale. Dans ce cas, on peut utiliser la « règle des 66-95-99,7 » [80], qui indique qu'environ 66% des valeurs se situent à un écart-type de la moyenne, 95% à deux écarts-types et 99,7% à trois écarts-types. Une valeur qui se situerait au-delà de trois écart-types de la moyenne aurait donc moins de 0,3% de chances d'être générée par cette distribution. Ainsi, pour chaque mesure s sur les données, on mesure la moyenne μ_s et l'écart-type σ_s sur le modèle nul, puis on mesure la crédibilité de ce modèle sur les données par le

biais du z-score :

$$z\text{-}score(s) = \frac{s - \mu_s}{\sigma_s} \tag{3.1}$$

Si le z-score dépasse trois en valeur absolue, je suppose que le modèle nul n'explique pas la valeur de la mesure.

3.1 Adaptation des motifs aux périodes d'activité

Je traite des réseaux dans lesquels les communications sont instantanées, j'adopte donc le formalisme de flux de liens. Comme Kovanen *et al.* [56], je prends en compte l'ordre des évènements dans les motifs.

Je m'intéresse aux réseaux de communications au sein desquels les individus ne sont pas forcément conscients de la réception d'un message. Il s'agit typiquement de réseaux d'e-mails ou des réponses à des messages sur des forums en ligne. Dans ce cas, le lien causal entre deux communications n'est pas clairement lié au délai de réponse. En effet, un individu peut mettre un temps important à réagir à un message du fait qu'il n'en ait pas conscience. Ma définition des motifs, les a-motifs, prend en compte cette spécificité. Elle correspond à une approche qui détecte plus de structures que celle définie par Zhao *et al.* [106].

Je sépare l'activité d'un individu en *périodes d'activité*. Ces périodes sont des intervalles de temps où un individu émet des messages en un délai court.

Définition 3.4 (μ-période d'activité)**.** *Pour chaque noeud $v \in V$ d'un flux de liens $G = (V, E)$, je note E_v l'ensemble des messages émis par v et $med(v \in V)$ la médiane du temps écoulé entre deux messages émis par un individu v sur l'ensemble du graphe. Je note aussi $t((u, v, x) \in E) = x$ la fonction associant la date d'un arc à l'arc lui-même. Une* ***μ-période d'activité*** *d'un individu $v \in V$ est un intervalle de temps $[a; b]$ durant lequel v a émis un ensemble de messages $M(a,b) = \{e \in E_v \mid a \leq t(e) \leq b\}$, qui a les propriétés :*

- $\exists e_1 \in M(a,b),\ t(e_1) = a$ *et* $\exists e_2 \in M(a,b),\ t(e_2) = b$ *et*
- $\forall e_1 \in M(a,b),\ t(e_1) \neq b \Rightarrow \exists e_2 \in M(a,b),\ 0 < t(e_2) - t(e_1) \leq \mu \cdot med(v)$ *et*
- $\forall e \in E_v,\ t(e) < a \Rightarrow t(e) < a - \mu \cdot med(v)$ *et* $t(e) > b \Rightarrow t(e) > b + \mu \cdot med(v)$.

C'est-à-dire que les messages émis pendant une période d'activité de v ne sont pas éloignés de plus de $\mu \cdot med(v)$ deux à deux et ces périodes sont maximales. Pour chaque individu v l'ensemble E_v de ses messages émis est une suite de μ-périodes d'activité. Je note $act(v,t)$ la date de fin de la dernière période d'activité avant un temps t d'un noeud.

Je définis maintenant les a-motifs de la même manière que les motifs, mais en prenant en compte les périodes d'activité. Soit $(u,v,t_1,t_2) \in E$ l'un des arcs d'une instance d'un a-motif. Les arcs émis par u ou par v avant la fin de leur prochaine période d'activité peuvent aussi être inclus dans cette instance.

Définition 3.5 (Instance de a-motif)**.** *Une instance de a-motif est un flux de liens $G = (V,E)$ tel que tout arc $e = (u,v,t)$ est soit :*

- *temporellement le premier du flux, $\forall e' \in E,\ t(e') > t(e)$ ou*
- *$\exists w \in V,\ t' \in \mathbb{R}^+,\ (u,w,t') \in E$ tel que $act(u,t) < t' < t$ ou*
- *$\exists w \in V,\ t' \in \mathbb{R}^+,\ (v,w,t') \in E$ tel que $act(v,t) < t' < t$.*

La fonction d'équivalence définissant les a-motifs est la même que celle des motifs. La détection des instances des a-motifs est illustrée Fig. 3.2.

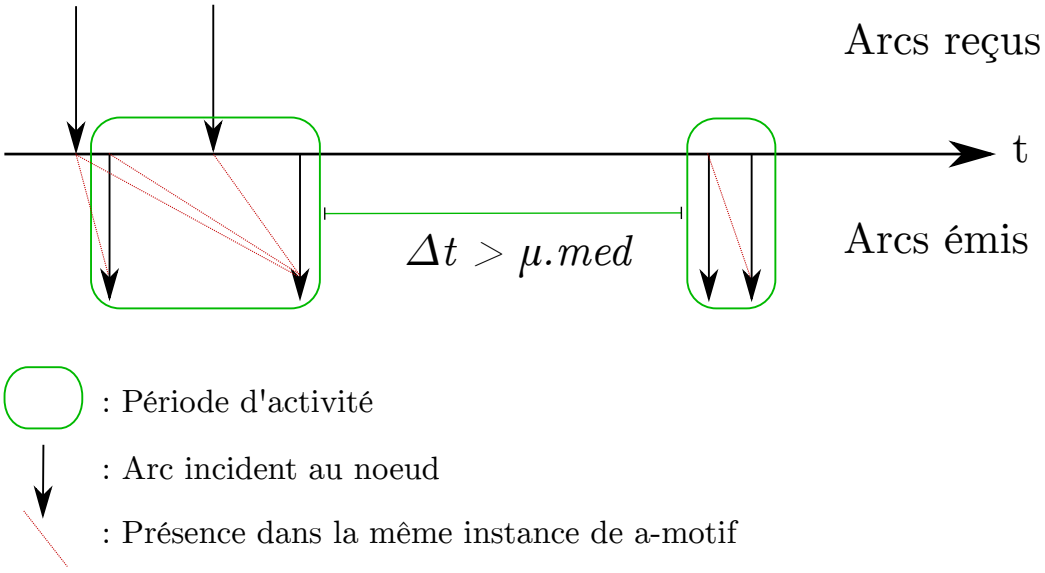


FIGURE 3.2 – Pour un noeud, les arcs émis forment des périodes d'activité. L'ensemble des arcs incidents forme des instances de a-motifs

Pour des raisons combinatoires, je me restreins aux a-motifs de taille 3, c'est-à-dire ceux qui contiennent 3 arcs. Je choisis cette taille car il s'agit

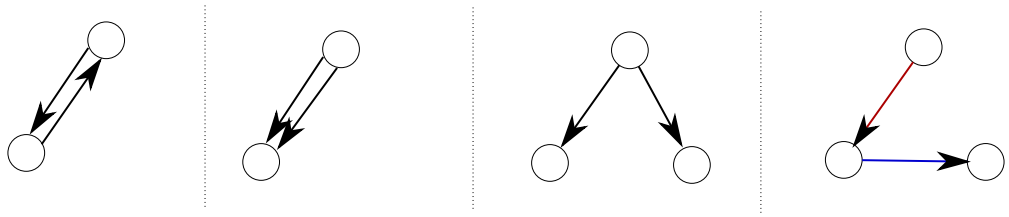


FIGURE 3.3 – Les motifs de taille 2. Quand l'ordre importe, l'arc rouge a une date antérieure à l'arc bleu.

d'un bon compromis entre le temps de calcul nécessaire à la détection des instances des a-motifs et la complexité des structures à observer.

Remarquons que les a-motifs de taille fixe sont en nombre fini. Par exemple, il existe 4 a-motifs de taille 2 (illustrés Fig. 3.3), et 26 a-motifs de taille 3.

Certaines périodes d'activité incluent des dizaines de communications tandis que d'autres n'en incluent qu'une. Si une période d'activité contient k arcs et que le noeud associé a reçu l arcs avant cette période, alors $k \cdot l$ instances de a-motifs de taille 2 seront générés. Cela signifie qu'un message reçu avant une période d'activité incluant beaucoup de messages a plus d'impact qu'un message reçu avant une période d'activité de petite taille.

Le choix que je fais dans ces travaux est de ne pas favoriser un message plus qu'un autre en raison de la taille des périodes d'activité. Pour cela, j'associe une pondération aux instances de telle sorte que la somme des poids de l'ensemble des instances générées par un seul arc source donne 1. Ce poids est calculé de la manière suivante : à partir d'une instance de poids p, si cette instance peut être étendue pour générer k instances de taille supérieure, chacune de ces instances a un poids p/k. Ainsi, si un arc génère originellement k_1 instances de taille 2, chacune a un poids $1/k_1$. Le premier de ces motifs génère alors k_2 instances de taille 3, chacune d'entre elles a un poids $1/(k_1 \cdot k_2)$. Le second des motifs de taille 2 génère k'_2 instances, chacune a un poids $1/(k_1 \cdot k'_2)$, et ainsi de suite. Chaque mesure utilisée dans les expériences de ce chapitre est pondérée par cette valeur.

Certains motifs de taille 3 sont géométriquement assez similaires entre eux, comme "AB-AC-BC" et "AB-AC-CB", ou "AB-BC-CB" et "AB-BA-BC". Afin de diminuer le nombre d'observations, je me concentre sur quatre motifs qui ont été identifiés comme importants dans la littérature associée [106, 89, 104]. Il s'agit de l'étoile "AB-AC-AD", du va-et-vient "AB-BA-AB", du triangle "AB-BC-CA" et de la chaine "AB-BC-CD". J'ajoute le a-motif de spam "AB-AB-AB".

3.2 Experiences

Aucun jeu de données ne fournit de vérité de terrain d'un graphe temporel. Il s'agit pourtant d'une donnée indispensable pour étudier le lien entre la structure communautaire et la façon dont les individus communiquent. Pour pallier ce manque, j'ai obtenu les données correspondant aux e-mails ayant transité sur les serveurs de l'Université de Caen. J'en ai extrait le réseau de communication correspondant aux échanges pendant 3 mois, accompagné de partitions provenant de l'annuaire.

Il existe trois données permettant d'établir une partition du réseau. Des utilisateurs sont associés à une entité de rattachement, typiquement une Unité de Formation et de Recherche ou un Institut Universitaire Technologique. Le laboratoire de rattachement de chaque chercheur est aussi identifié. Enfin, la section CNU (Conseil National des Universités) est renseignée, ce qui associe l'utilisateur à un domaine d'étude. En règle générale, un étudiant dépend d'une entité et d'une section CNU, un chercheur est en plus affecté à un laboratoire et un personnel de support peut n'être attaché qu'à une entité.

Le réseau résultant a les caractéristiques suivantes :

— 7 688 665 messages échangés entre 210 085 adresses,

— 1 275 662 messages échangés entre 26 177 adresses ayant une entité de rattachement renseignée,

— 168 507 messages échangés entre 918 adresses appartenant à un laboratoire,

— 378 721 messages échangés entre 17 275 adresses ayant une section CNU renseignées.

Sont représentées 57 entités de rattachement (459 adresses par entité en moyenne), 45 laboratoires (20 adresses en moyenne) et 146 sections CNU (118 adresses par section en moyenne). Plus de détails sur le contenu du réseau et les démarches qui ont étés nécessaires pour l'obtenir sont disponibles App. A. Puisqu'il y a trois partitions distinctes, je crée trois flux de liens à partir de ce réseau. Chacun de ces flux de liens correspond aux communications entre les individus ayant l'une de ces informations indiquée.

En plus du réseau de mails de l'université de Caen, j'utilise un ensemble des réseaux de communication provenant du site web Konect [1]. Le seul traitement effectué sur ceux-ci est la suppression des boucles et des noeuds de degré nuls.

1. `http://konect.uni-koblenz.de`

Nom	n	m	noeuds	arcs
Enron [55]	86978	1134990	employés	e-mails
Facebook [97]	45813	855542	utilisateurs	écritures sur les « murs »
UC Irvine [76]	1899	59835	étudiants	messages
Radoslaw [69]	167	82876	employés	e-mails
Debian [38]	34648	316569	utilisateurs	réponses
Digg [29]	30360	86203	utilisateurs	réponses
Linux Kernel Mailing List (LKML)[2]	26885	1028233	utilisateurs	réponses
Slashdot [44]	51083	139789	utilisateurs	réponses

TABLE 3.1 – Les réseaux de Konect

Aucun des réseaux de Konect n'est associé à une vérité de terrain. Je génère une partition de ces flux de liens à partir de l'algorithme de Louvain [9]. Cet algorithme s'applique sur des graphes non temporels. Je génère donc des graphes non temporels en utilisant les mêmes noeuds que le graphe temporel et un lien entre deux noeuds si le graphe temporel contient une interaction entre ceux-ci.

Ces communautés sont donc générées en ignorant l'aspect temporel des réseaux. Mon objectif est de déterminer si les communautés ainsi générées et les communautés observées ont des propriétés similaires en ce qui concerne les motifs.

Je classe ces réseaux en trois catégories :

Les plateformes de communication directe où les utilisateurs s'envoient des messages. L'entreprise Enron a dû, au terme d'un procès, publier les e-mails qui ont transité par ses serveurs. Le réseau Facebook représente la publication sur les « murs » dans le réseau social web, c'est-à-dire des messages publics entre utilisateurs. UC Irvine contient les messages transmis entre les membres d'une communauté étudiante de l'« University of California ». Radoslaw correspond à la communication interne entre employés d'une entreprise de manufacture.

Les listes de diffusion techniques où les utilisateurs posent et répondent à des questions. Deux listes de diffusions sont présentées, celle de Debian et celle du noyau Linux (LKML)

Les sites d'informations où les utilisateurs commentent un sujet. Les utilisateurs du site Digg peuvent proposer, voter et commenter des actualités.

2. `http://konect.uni-koblenz.de/networks/lkml-reply`

Le site Slashdot fonctionne de manière similaire et a une orientation technophile.

Les réseaux provenant de l'université de Caen appartiennent à la première catégorie.

3.2.1 Analyse des propriétés des a-motifs

Tout d'abord, j'analyse les statistiques de base des a-motifs afin de déterminer les différences avec les motifs. Dans ces expériences, j'ai supposé qu'une période d'activité englobait typiquement plusieurs communications. Pour $\mu = 1$ (voir définition 3.4), la moitié des arcs termine une période d'activité, ce qui implique que beaucoup de périodes ne contiennent qu'un arc dans les jeux de données. Afin d'obtenir des périodes d'activité correspondant mieux à mes suppositions, j'ai choisi $\mu = 2$.

Zhao *et al.* [106] ont observé que les motifs de chaine et d'étoile sont les plus fréquents. C'est aussi ce que l'on observe Fig. 3.4. Les a-motifs de chaîne sont les plus fréquents (16% en moyenne), suivis par l'étoile (6%) et le va-et-vient (3%).

Je mesure ensuite le z-score de ces fréquence, présenté Fig. 3.5. Je vais comparer les observations de Tabourier *et al.* [89] sur un réseau d'appels téléphoniques avec mes observations sur les réseaux mentionnés plus haut. Comme relevé dans [89], les a-motifs d'étoile sont moins communs dans les réseaux réels que dans le modèle nul, ce qui se traduit dans mes mesures par un z-score négatif. Les a-motifs de chaine ont un z-score négatif, ce qui est l'inverse de ce que Tabourier *et al.* avaient relevé. Une explication possible de cette différence est que la définition des a-motifs entraine l'apparition de beaucoup de chaines dans le modèle nul. En effet, si la fenêtre temporelle est remplacée par des périodes d'activité, une chaine peut être beaucoup plus facilement initiée par des émissions aléatoires.

Les z-scores mesurés ici sont hauts. Ceci implique une différence importante avec le modèle nul, la causalité semble donc avoir une influence importante sur les a-motifs relevés.

Sur l'ensemble des réseaux considérés, le spam et le va-et-vient ont des z-scores toujours positifs tandis que l'étoile a un z-score toujours négatif. Je rejoins l'observation de Zhao *et al.* : le va-et-vient a le z-score le plus important. J'apporte une nouvelle information : le a-motif de spam, qui n'a pas été considéré dans la littérature, présente une fréquence non expliquée par le modèle nul. Je mesure que les a-motifs d'étoile sont statistiquement plus présents dans le modèle nul que dans les données. J'en déduis que les a-motifs d'étoile des données sont majoritairement dues au hasard.

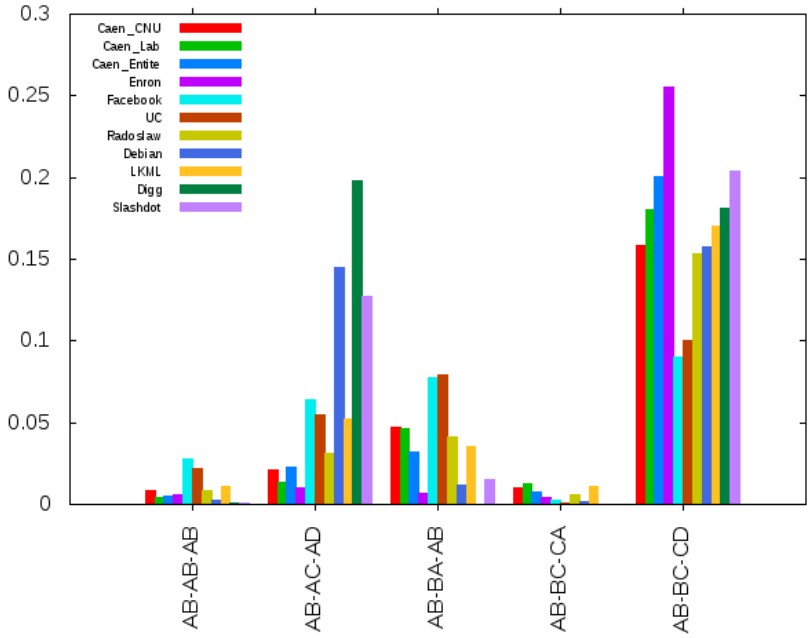


FIGURE 3.4 – La fréquence des a-motifs des différents réseaux

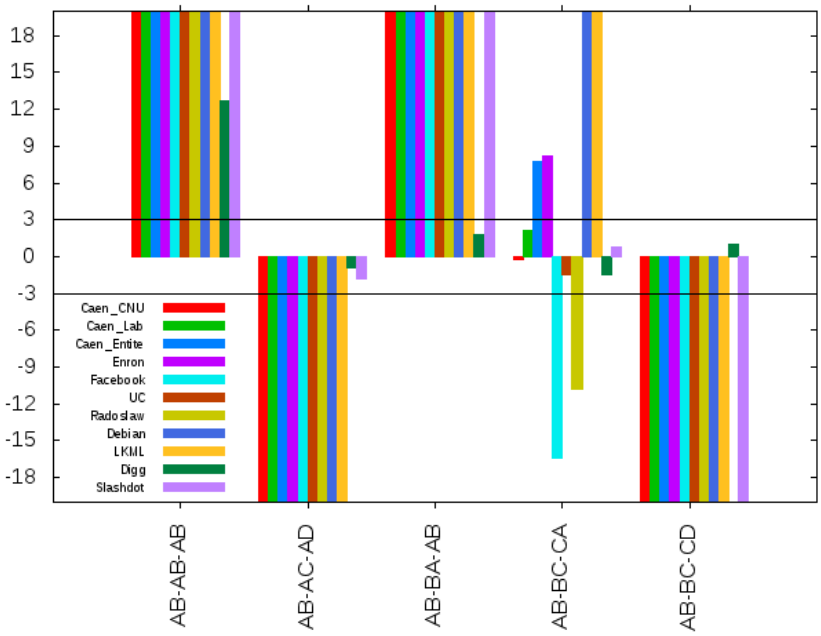


FIGURE 3.5 – Le z-score de la fréquence des a-motifs des différents réseaux. Un score dépassant 3 en valeur absolue est considéré comme très significatif. Les valeurs supérieures à 20 ou inférieures à -20 sont tronquées.

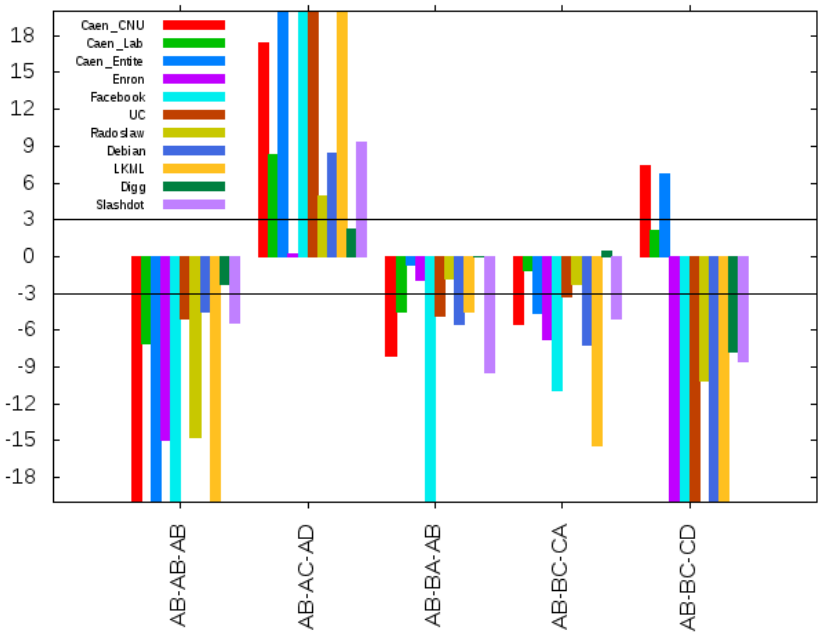


FIGURE 3.6 – Le z-score de la durée entre l'émission du premier message et du second

Le a-motif de triangle ne fait pas consensus au niveau du z-score des jeux de données. Ce a-motif a aussi le z-score moyen le plus faible parmi les a-motifs observés. Même si les z-scores du triangle sont en moyenne positifs, l'influence de la causalité sur la fréquence de ces motifs n'est pas claire de manière générale.

Une autre indication que les a-motifs sont similaire aux motifs est présentée Fig. 3.6. En effet, pour la plupart des a-motifs, la durée entre l'émission des messages est inférieure à celle du modèle nul. Ceci indique un enchainement plus rapide des communications que ce que prévu par le modèle. Comme il s'agit du critère utilisé pour définir les motifs, les a-motifs semblent similaires aux motifs.

Le a-motif d'étoile est le seul dont la durée entre les émissions a un z-score positif. L'intervalle de temps entre les deux premiers messages des instances du a-motif étoile est donc plus long dans les données que dans le modèle nul.

Les jeux de données provenant de l'université de Caen ont un z-score positif sur la durée du a-motif chaine, contrairement à tous les autres jeux de données. Il est notable que ces jeux de données couvrent des vacances scolaires d'une à deux semaines suivant les établissements. Il est possible qu'un grand nombre d'e-mails ait été envoyé durant ces vacances, et que ces

e-mails aient formé des chaines. Le délai important de réponse serait alors dû aux vacances.

Je conclue que certains a-motifs sont influencés par la causalité. En effet, les mesures effectuées montrent une différence importante des propriétés des instances de ces a-motifs entre le modèle nul et les jeux de données.

On a pu voir que les instances des a-motifs de spam et de va-et-vient sont plus fréquentes et plus courtes que ces mêmes instances dans le modèle nul. J'interprète ces résultats comme dus à un lien causal entre les instances de ces motifs.

Les instances du a-motif d'étoile ont des propriétés inverses : dans les données, elles sont moins fréquentes et plus longues que dans le modèle nul. Or, dans ce modèle où la causalité est supprimée, les a-motifs tels que le spam et le va-et-vient deviennent moins fréquents. Je suppose alors que le modèle forme des instances de a-motifs d'étoile avec les arcs qui composaient les instances des autres a-motifs dans les données.

Les instances du a-motif de chaine dans les données sont moins fréquentes et plus courtes que dans le modèle nul. Ces différences suggèrent que les a-motifs de chaine observés sont liés causalement.

Il n'y a pas de consensus quant à la différence de fréquence du motif de triangle sur les jeux de données. Les z-scores de la durée sont cependant majoritairement négatifs, ce qui laisse supposer que les triangles sont liés causalement.

3.2.2 Etude de la relation entre a-motifs et communautés

J'étudie ensuite les arcs utilisés par les a-motifs, en distinguant arcs internes aux communautés et arcs externes. L'objectif de cette expérience est de comparer les propriétés des a-motifs se produisant entre les communautés et ceux se produisant à l'intérieur. Soit $w_{in}(m)$ la somme des poids des a-motifs de type m qui utilisent un arc interne. On s'intéresse aux poids relatifs, donc $w_{in}^{norm}(m) = w_{in}(m) / \sum_{m' \in M} w_{in}(m')$ avec M l'ensemble des a-motifs extraits (ici, l'ensemble des a-motifs de taille 3). La distribution de ces poids est similaire à la Fig. 3.4.

Je calcule un ratio entre ces poids internes et les poids externes décrit Eq. 3.2.

$$ratio(m) = \frac{w_{ext}^{norm}(m) - w_{in}^{norm}(m)}{\max(w_{ext}^{norm}(m), w_{in}^{norm}(m))} \tag{3.2}$$

Si ce ratio dépasse zéro pour un a-motif, les instances de cet a-motif ont une

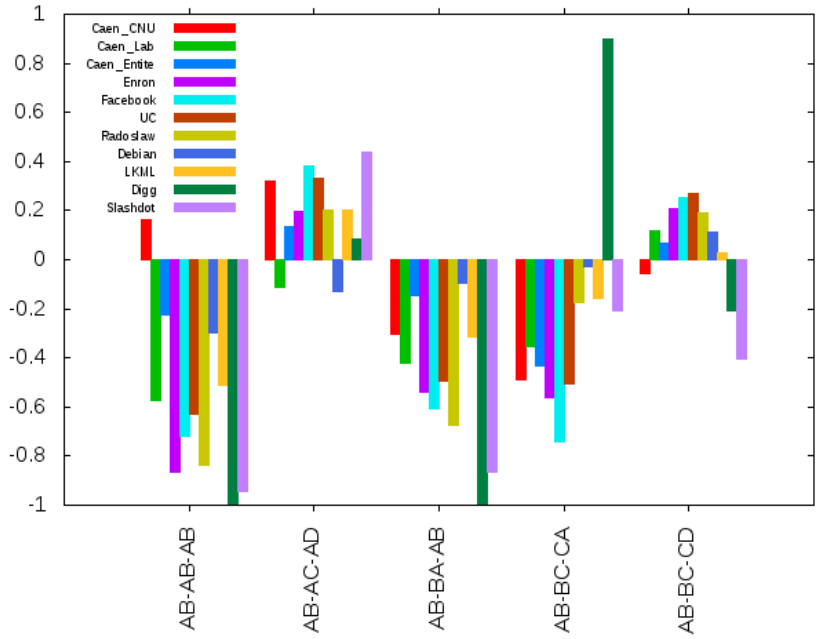


FIGURE 3.7 – Le ratio entre la proportion extérieure et intérieure des a-motifs

fréquence relative supérieure à l'extérieur des communautés par rapport à l'intérieur. Les résultats sont présentés Fig. 3.7.

Tout d'abord, le a-motif de va-et-vient semble avoir une fréquence relative plus importante à l'intérieur des communautés, tous les jeux de données ayant un $ratio(\text{AB-AB-AB}) < 0$. Le a-motif de spam présente les mêmes caractéristiques, sauf pour le réseau de Caen_CNU qui a un ratio au-dessus de zéro (ratio de 0,16). De même, les ratios du a-motif de triange sont négatifs sauf pour Debian (ratio de 0,9). L'inverse se produit pour le a-motif d'étoile, majoritairement plus fréquent sur la frontière des communautés, sauf pour Caen_Lab (ratio de $-0{,}11$), et Debian (ratio do $-0{,}13$). Pour le a-motif chaine, les ratios sont majoritairement positifs mais trois des jeux de données ont des ratios négatifs. Il est donc difficile d'en tirer une conclusion générale.

La Figure 3.8 montre que la grande majorité ($\sim 84\%$) des z-scores du ratio précédemment étudié sont positifs. Ceci implique que les ratios dans les jeux de données sont supérieurs à ceux du modèle nul. Donc, dans le jeu de données, les a-motifs se situent plus souvent à l'extérieur des communautés par rapport à ce que le modèle nul prévoyait.

Je conclue que ces résultats montrent une tendance assez claire pour chaque a-motif à se dérouler soit exclusivement à l'intérieur des communautés soit

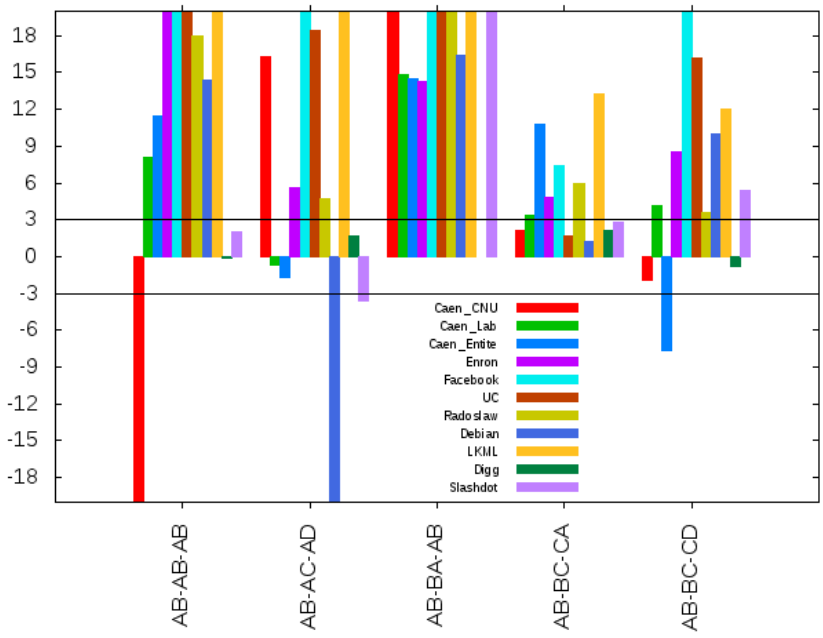


FIGURE 3.8 – Le z-score du ratio Fig. 3.7

exclusivement à l'extérieur. D'un côté, les instances des a-motifs de chaîne et d'étoile sont majoritairement externes. D'un autre côté, les instances des a-motifs de spam, de va-et-vient et de triangle sont majoritairement internes. Il semble donc que les motifs puissent servir à différencier arcs internes et externes des communautés. Il y a donc des modes de communication différents à l'intérieur des communautés et à l'extérieur. Dans le premier cas, on retrouve plutôt des dialogues, comme l'indique les motifs de va-et-vient et de triangle. Dans le second cas, on identifie des motifs de propagation d'information non réciproques.

Les résultats montrent que la causalité a pour effet que les motifs sont plus externes que prévu par le modèle nul. Une interprétation possible de ce résultat est que c'est sur la frontière des communautés que se déroule la plus grande partie des a-motifs liés par la causalité. Une explication est que les communications qui se déroulent aux frontières ont besoin de structures particulières. En effet, les individus qui s'y transmettent des informations servent de point de transfert entre les deux communautés, ce qui génère des a-motifs spécifiques quand ils sont diffusés.

Enfin, on remarque que les communautés produites par l'algorithme de Louvain et celles provenant des jeux de données de l'Université de Caen présentent des résultats similaires. Cela implique que les propriétés tempo-

relles observées sont des caractéristiques de communautés structurelles. Ces observations ne différencient pas les communautés observées sur un réseau temporel des communautés obtenues par un graphe.

Dans ce chapitre, j'ai présenté une définition alternative des motifs dans un graphe temporel qui prend en compte les périodes d'activité dans les communications. Avec l'aide de données fournies par l'Université de Caen et avec d'autres données de la littérature, j'ai pu établir certaines propriétés de ces motifs. Je relève notamment des différences importantes entre les motifs relevés sur des données réelles et les motifs d'un modèle nul qui a été créé pour supprimer la causalité dans le réseau. Cela porte à croire que cette version détecte des motifs liés causalement.

J'ai aussi observé le lien entre la structure communautaire et les motifs. J'ai pu remarquer que certains motifs, comme le spam, le va-et-vient et le triangle, étaient généralement plus présents à l'intérieur qu'à l'extérieur des communautés. Un autre motif, l'étoile, apparait fréquemment à l'extérieur des communautés. En plus, tous les motifs apparaissent plus fréquemment à l'extérieur qu'à l'intérieur des communautés par rapport à ce que le modèle nul prévoyait. Ces observations pourraient aider le développement de la détection de communautés sur flux de liens, en étant utilisé en tant que fonction de qualité, par exemple.

Chapitre 4

Une nouvelle fonction de qualité : la compacité

Dans ce chapitre, je définis une fonction de qualité, la compacité. Celle-ci est basée sur un modèle de diffusion où chaque individu retransmet directement les informations reçues à chaque pas de temps. Dans ce modèle, une information se diffuse rapidement dans une communauté si les chemins entre ses membres sont courts. J'étudie ensuite cette mesure de manière analytique.

Les travaux présentés dans ce chapitre ont fait l'objet de deux publications [25, 26].

Dans le chapitre précédent, j'ai établi qu'il était possible de différencier arcs internes et arcs externes des communautés en observant l'ordre des communications entre les membres du réseau.

On sait que la diffusion d'information est particulièrement rapide au sein de certaines communautés comme pour les utilisateurs de forums de santé [15]. Une faible longueur des chemins accélère la diffusion d'information [24]. Par conséquent, mesurer la taille des chemins entre les membres d'une communauté caractérise en partie la vitesse de diffusion des messages au sein de ses membres.

Ce critère de la distance dans les communautés n'est pas directement corrélé au critère de séparabilité (voir Sec. 2.3). Leskovec *et al.* [64] comparent les « clusters » résultants de l'application de deux algorithmes de détection de communautés sur les mêmes graphes. Ils observent notamment que les « clusters » produits par l'un des deux algorithmes présentent des chemins plus courts que ceux de l'autre. Cependant, les « clusters » ayant des chemins courts ont aussi une moins bonne conductance que ceux produits par l'autre algorithme.

Dans ce chapitre, je propose une nouvelle fonction de qualité appelée compacité. Elle se base sur le diamètre pour estimer l'efficacité d'une diffusion d'information dans une communauté. Je montre ensuite que cette fonction de qualité satisfait les propriétés de Van Laarhoven et Marchiori [95], conçues pour décrire un comportement intuitif des fonctions de qualité.

Alors que je note la compacité $q_{comp}(C)$ dans le reste du manuscrit, je la désigne par $\texttt{Comp}(C)$ dans ce chapitre afin d'alléger les notations.

4.1 Modélisation et définition d'une structure communautaire compacte

Dans cette section, je présente une nouvelle fonction de qualité, la compacité.

Certaines communautés permettent une diffusion rapide des informations [15]. La compacité permet d'identifier les communautés où la diffusion d'information est rapide en mesurant la vitesse de propagation de celles-ci. Le terme « compacité » provient du fait que les structures favorisées par ce type de mesure ont un diamètre faible et sont donc « compactes ».

J'utilise l'« Independant Cascade Model » [40, 53] avec une probabilité de contamination de 1 comme modèle de diffusion d'information. Dans ce modèle, un unique individu possède initialement une information. À chaque pas de temps, chaque individu possédant l'information la transmet à ses voisins qui ne la possèdent pas, en activant l'arête qui les relie. On remarque alors que le temps nécessaire à ce que la diffusion atteigne toute la communauté est égal à la distance maximale entre le noeud d'origine et les autres noeuds de la communauté. La vitesse de propagation d'une diffusion d'information correspond au nombre moyen d'arêtes activées par pas de temps lors de la diffusion.

Le noeud de départ d'une diffusion a une influence sur la vitesse de propagation En effet, une diffusion partant d'un noeud proche de tous les autres dans la communauté a une vitesse de propagation bien plus importante qu'en partant d'un noeud excentré. Je considère la pire vitesse de propagation dans le pire des cas. Ce choix permet de garantir que les diffusions dans les communautés « compactes » sont toujours rapides. Le pire des cas a lieu quand le noeud de départ d'une diffusion se situe à l'extrémité d'un diamètre.

L'ensemble de ces éléments amène à la définition formelle de la compacité.

Définition 4.1 (Compacité). *La compacité d'un « cluster » c est définie par :*

$$\textit{Comp}(c) = \begin{cases} 0 & \text{si } diam(c) = 0 \text{ ou si } c \text{ non connexe} \\ \dfrac{m(c)}{diam(c)} & \text{sinon} \end{cases}$$

Pour calculer la compacité d'une partition, on somme la compacité des « clusters » qui la composent.

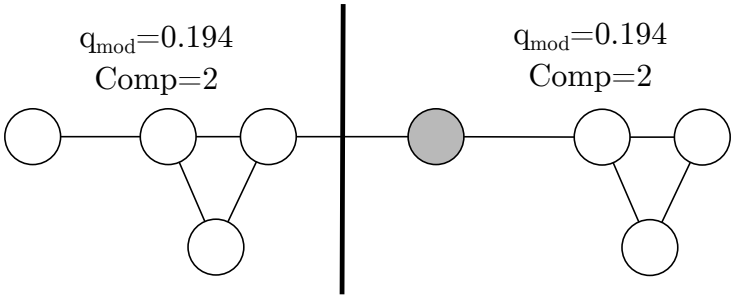
Le désavantage de l'utilisation du diamètre est que la complexité de sa mesure est quadratique en la taille de la communauté. De plus, le calcul doit être refait entièrement en cas de modification de la partition. Je note donc que la compacité est plus adaptée à l'évaluation de partitions, qui peut être faite sur des réseaux de taille réduite, qu'à de l'optimisation gloutonne.

La compacité n'utilisant que les arcs internes, elle n'est pas affectée par les relations des membres de la communauté avec l'extérieur de celle-ci. Ainsi, tout ensemble de noeuds est optimal en compacité si ses noeuds forment une clique.

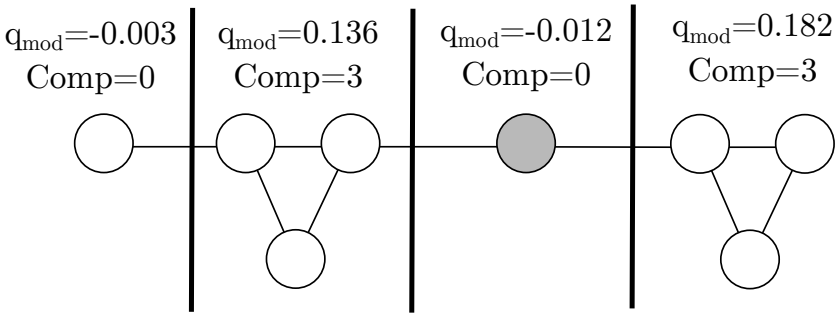
La compacité et les fonctions se basant sur la séparabilité ne favorisent pas les mêmes « clusters ». Pour l'illustrer, je compare la compacité et la modularité sur des exemples jouet. J'appelle informellement « satellite » un noeud de degré 1 et « pont » un noeud à la jonction de sous-parties denses du graphe, comme le noeud grisé Fig. 4.1a.

La modularité a tendance à favoriser les « clusters » qui incluent des satellites et des ponts (par exemple Fig. 4.1). Sur cet exemple, la partition présenté Fig. 4.1a a une modularité plus importante qu'en Fig. 4.1b, où le satellite ainsi que le pont sont dans leur propre « cluster ». La modularité peut donc être utilisée dans une application où chaque noeud doit être classé dans une communauté non triviale. Néanmoins, cette approche n'est pas adaptée quand le but est de trouver des communautés où la transmission d'informations est efficace. La compacité correspond à une approche plus conservatrice : si un utilisateur n'a qu'une seule connexion, il semble difficile de conclure sur son appartenance communautaire.

Un autre exemple soulignant la différence entre les deux mesures est présenté Fig. 4.2. La partition optimale en modularité sépare les noeuds de la 3-clique centrale pour grouper les satellites connectés aux noeuds participant à la clique. D'un autre côté, la compacité favorise la cohésion interne, par conséquent son optimum regroupe la 3-clique centrale et isole les satellites. On peut étendre l'exemple en considérant une k-clique centrale où chaque noeud est de plus connecté à $k + 1$ satellites. Pour tout $k > 2$ entier, la modularité favorisera systématiquement le regroupement avec les satellites tandis que la compacité favorisera la conservation de la clique centrale.



(a) Partition incluant noeuds satellites et de connexion



(b) Partition excluant noeuds satellites et de connexion

FIGURE 4.1 – Modularité and compacité de deux partitions du même graphe

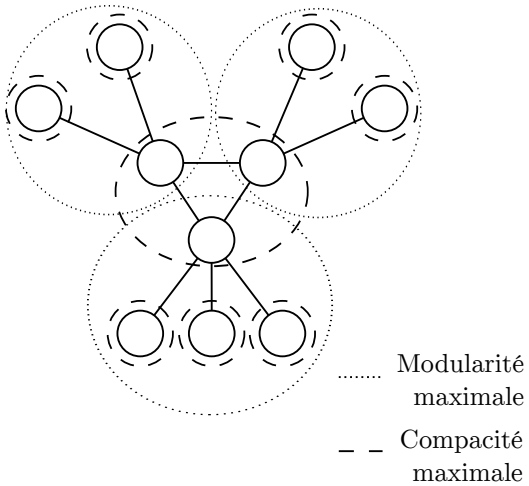


FIGURE 4.2 – Un cas simple d'une clique centrale avec des satellites.

4.2 Respect des axiomes de Van Laarhoven et Marchiori

Dans cette section, je montre que la compacité respecte un ensemble de propriétés conçues pour représenter un comportement intuitif de la communauté.

Van Laarhoven et Marchiori [95] ont créé un ensemble de six axiomes : **invariance par permutation**, **invariance d'échelle**, **richesse**, **monotonicité**, **localité** et **continuité**. Selon eux, une fonction de qualité devrait respecter ces axiomes dans le contexte du partitionnement de graphes. Ces axiomes sont différents des propriétés présentées Sec. 2.3. En effet, ces propriétés sont utilisées dans les définitions des communautés. Les axiomes de Van Laarhoven et Marchiori sont indépendants des définitions des communautés.

Dans leur article [95], les auteurs prouvent notamment que la modularité ne respecte pas les axiomes de **localité** et de **monotonicité**. Ils ne présentent aucune fonction de qualité sans paramètre satisfaisant tous ces axiomes.

Ces propriétés portent sur des graphes pondérés. Je définis maintenant la compacité sur les graphes pondérés. La communication entre deux individus proches permet souvent une transmission rapide des informations entre ces individus [53]. Afin d'utiliser le même formalisme que Van Laarhoven et Marchiori [95], un poids fort sur une arête signifie une grande affinité donc une vitesse de transmission faible. Utiliser la distance classique d'un graphe pondéré (c'est-à-dire la somme des poids des arêtes) produit un résultat contraire à ce que je souhaite obtenir. Je considère donc le poids associé aux arêtes comme étant une vitesse de transmission, l'inverse de celui-ci étant alors le temps de transmission. Dans un graphe pondéré (V, w), la fonction de longueur d'un chemin π que j'utilise est la suivante :

$$len(\pi) = \sum_{e \in \pi} \frac{1}{w(e)} \tag{4.1}$$

Cette définition de la longueur d'un chemin est utilisée par la compacité dans le calcul du diamètre. Le numérateur correspond à la somme du poids des arêtes internes au « cluster » considéré.

Je vais maintenant montrer que la compacité respecte les six axiomes de Van Laarhoven et Marchiori.

Définition 4.2 (Invariance par permutation)**.** *Une fonction de qualité Q est **invariante par permutation** si, pour tous graphes $G = (V,w)$, toute*

partition $C \in \mathcal{C}(V)$ et tout isomorphisme f qui à G associe $f(G) = (V', w')$, on a $Q_G(C) = Q_{f(G)}(f(C))$.

Intuition : Une fonction de qualité ne devrait pas dépendre de la représentation du graphe.

Théorème 4.3. *La compacité est invariante par permutation.*

Démonstration. Par abus de notation, j'utilise f sur des « clusters » et des noeuds. Prouvons que les distances sur les graphes pondérés sont invariantes par permutation, en utilisant la notion de longueur définie Eq. 4.1 :

$$\begin{aligned} dist_{f(G)}(f(u), f(v)) &= \min_{\pi \in \Pi_{f(G)}(f(u),f(v))} (len_{f(G)}(\pi)) \\ &= \min_{\pi \in \Pi_G(u,v)} (len_G(f^{-1}(\pi))) \\ &= dist_G(u, v) \end{aligned}$$

Ceci implique que la compacité est invariante par permutation :

$$\begin{aligned} \texttt{Comp}_{f(G)}(f(C)) &= \sum_{f(c) \in f(C)} \frac{\sum_{(f(i),f(j)) \in f(c)^2} w'(f(i), f(j))}{\max_{(f(u),f(v)) \in f(C)^2} (dist_{f(G)}(f(u),f(v)))} \\ &= \sum_{f(c) \in f(C)} \frac{\sum_{(f(i),f(j)) \in f(c)^2} w(i, j)}{\max_{(f(u),f(v)) \in f(c)^2} (dist_G(u, v))} \\ \texttt{Comp}_{f(G)}(f(C)) &= \texttt{Comp}_G(C) \end{aligned}$$

La compacité est stable par isomorphisme, et est donc invariante par permutation. □

Définition 4.4 (Invariance en échelle)**.** *Une fonction de qualité Q est **invariante en échelle** si pour tous les graphes $G = (V, w)$, toutes les partitions $C_1, C_2 \in \mathcal{C}(V)$ de G et toutes les constantes $\alpha > 0$, $Q_G(C_1) \leq Q_G(C_2)$ si et seulement si $Q_{\alpha G}(C_1) \leq Q_{\alpha G}(C_2)$, où $\alpha G = (V, \alpha \cdot w)$ est le graphe G dont les poids des arêtes est multiplié par α.*

Intuition : Le poids des arêtes a souvent une échelle choisie arbitrairement. Tout changement sans inversion de cette échelle ne devrait pas avoir d'impact sur le classement des partitions par la fonction de qualité.

Théorème 4.5. *La compacité est invariante en échelle*

Démonstration. Commençons par prouver une corrélation linéaire des longueurs des chemins entre G et αG.

$$\begin{aligned} len_{\alpha G}(v_0, ..., v_k) &= \sum_{i=0}^{k-1} \frac{1}{\alpha \cdot w(v_i, v_{i+1})} \\ len_{\alpha G}(v_0, ..., v_k) &= \frac{len_G(v_0, ..., v_k)}{\alpha} \end{aligned}$$

Du fait que les longueurs des chemins sont corrélés linéairement, les chemins minimums sont les mêmes dans G et dans αG, ce qui implique :

$$\begin{aligned} dist_{\alpha G}(u, v) &= \min_{\pi \in \Pi_{\alpha G}(u,v)} len_{\alpha G}(\pi) \\ &= \min_{\pi \in \Pi_{\alpha G}(u,v)} \frac{len_G(\pi)}{\alpha} \\ &= \frac{dist_G(u, v)}{\alpha} \end{aligned}$$

Par le même raisonnement, $diam_{\alpha G}(c) = diam_G(c)/\alpha$ La compacité peut être écrite de la manière suivante :

$$\begin{aligned} \texttt{Comp}_{\alpha G}(C) &= \sum_{c \in C} \sum_{(i,j) \in c^2} \frac{w_{\alpha G}(i, j)}{diam_{\alpha G}(c)} \\ &= \sum_{c \in C} \sum_{(i,j) \in c^2} \alpha^2 \frac{w_G(i, j)}{diam_G(c)} \\ &= \alpha^2 \texttt{Comp}_G(C) \end{aligned}$$

Par conséquent, pour toutes partitions $C_1, C_2 \in \mathcal{C}(V)$, si $\texttt{Comp}_G(C_1) \geq \texttt{Comp}_G(C_2)$ alors $\alpha^2 \texttt{Comp}_G(C_1) \geq \alpha^2 \texttt{Comp}_G(C_2)$, ce qui implique $\texttt{Comp}_{\alpha G}(C_1) \geq \texttt{Comp}_{\alpha G}(C_2)$. La compacité est donc invariante en échelle. □

Définition 4.6 (Richesse)**.** *Une fonction de qualité Q est **riche** si pour tous les ensembles V et toutes les partitions $C^* \in \mathcal{C}(V)$, il existe un graphe $G = (V, w)$ tel que C^* est la partition Q-optimale de V, c'est-à-dire que $C^* = \arg\max_{C \in \mathcal{C}(V)} Q_G(C)$.*

Intuition : N'importe quelle partition d'ensemble de noeuds devrait pouvoir être optimale par rapport à la fonction de qualité, si la structure du graphe correspond.

Théorème 4.7. *La compacité est riche*

Démonstration. $\forall C^* \in \mathcal{C}(V)$, soit $G = (V, w)$ un graphe tel que, $\forall (i,j) \in V^2$, $w(i,j) = 1$ si $\exists c \in C^*$ tel que $i \in c$ et $j \in c$ (i et j appartiennent au même « cluster ») et $w(i,j) = 0$ sinon. G est donc un graphe dont toutes les composantes connexes sont des cliques, et ces cliques sont les « clusters » composant C^*. Soit $C \in \mathcal{C}(V)$ une partition optimale en compacité de G, c'est-à-dire que $\arg\max_{D \in \mathcal{C}(V)} \texttt{Comp}_G(D) = C$.

Si $\exists c \in C$ tel que $\exists (i,j) \in c^2$, $w(i,j) = 0$, alors i et j ne sont pas dans le même « cluster » dans C^*. Donc, c est déconnecté, ce qui implique que sa compacité est nulle. Si c est composée de k composantes connexes, on crée k nouveaux « clusters » $(c_1, ..., c_k)$ contenant chacun les noeuds d'une composante connexe différente. Remarquons que la compacité de chacun de ces « clusters » est positive, et donc supérieure ou égale à la compacité de c. Soit $C' \in \mathcal{C}(V)$ telle que $C' = C \setminus c \cup \{c_1, ..., c_k\}$. Alors :

$$\begin{aligned} \texttt{Comp}(C') &= \texttt{Comp}(C \setminus c) + \texttt{Comp}(\{c_1, ..., c_k\}) \\ &\geq \texttt{Comp}(C \setminus c) + \texttt{Comp}(c) \\ \Leftrightarrow \texttt{Comp}(C') &\geq \texttt{Comp}(C) \end{aligned}$$

Ces scores sont égaux si et seulement si tous les noeuds de c ont un degré nul. Nous pouvons donc conclure que $\forall c \in C$, c est connexe ou composée uniquement de noeuds à degré nul (condition de connexité).

Soient $c_1, c_2 \in C, c_1 \neq c_2$ tels que $\exists i \in c_1$, $\exists j \in c_2$ avec $w(i,j) = 1$. Il y a donc une ou plusieurs arêtes entre deux « clusters » dans C. Alors i et j sont dans le même « cluster » dans C^* et dans différents « clusters » dans C. Les noeuds i et j ont un degré non nul, donc par la condition de connexité les « clusters » c_1 et c_2 sont tous deux connexes, ce qui implique que $c_1 \cup c_2$ est connexe. Nous appelons $C' \in \mathcal{C}(V)$ la partition correspondant à C où c_1 et c_2 ont étés remplacés par leur fusion, formellement $C' = (C \setminus \{c_1, c_2\}) \cup \{c_1 \cup c_2\}$.

$$\begin{aligned} \texttt{Comp}(C') &= \texttt{Comp}(C \setminus \{c_1, c_2\}) + \texttt{Comp}(\{c_1 \cup c_2\}) \\ &\geq \texttt{Comp}(C \setminus \{c_1, c_2\}) + \texttt{Comp}(c_1) + \texttt{Comp}(c_2) + \frac{w(i,j)}{diam(\{c_1 \cup c_2\})} \\ &> \texttt{Comp}(C \setminus \{c_1, c_2\}) + \texttt{Comp}(c_1) + \texttt{Comp}(c_2) \\ \Leftrightarrow \texttt{Comp}(C') &> \texttt{Comp}(C) \end{aligned}$$

Du fait que C soit optimal par rapport à `Comp`, il n'y a aucune arête entre les « clusters » de C, ce qui est équivalent à $\forall c \in C$, $\exists c' \in C^*$, $c' \subseteq c$ (condition de maximalité).

Les deux conditions réunies impliquent que tout « cluster » dans C est soit une composante connexe maximale de G (et donc un « cluster » dans C^*) ou un ensemble de noeuds de degré nul. Comme tout ensemble contenant un noeud de degré nul a la même compacité (zéro), C^* a la même compacité que toute autre partition optimale en compacité. Donc C^* est une partition de G de compacité maximale. La compacité est donc riche. □

Avant de définir la monotonicité, j'ai besoin de définir la notion *d'amélioration C-consistante* d'un graphe par rapport à une partition.

Définition 4.8 (Amélioration C-consistante)**.** *Soit $G = (V, w)$ un graphe et $C \in \mathcal{C}(V)$. Un graphe $G' = (V, w')$ est une* ***amélioration C-consistante*** *de G si pour tous les noeuds i et j, $w'(i,j) \geq w(i,j)$ quand i est dans la même communauté que j et $w'(i,j) \leq w(i,j)$ quand i n'est pas dans la même communauté que j.*

Définition 4.9 (Monotonicité)**.** *Une fonction de qualité Q est* ***monotone*** *si pour tous les graphes G, toutes les partitions $C \in \mathcal{C}(V)$ de G et toutes les améliorations C-consistantes G' de G, $Q_{G'}(C) \geq Q_G(C)$*

Intuition : La fonction de qualité ne peut pas contredire la supposition de base de la structure communautaire : ce sont des zones denses du graphe.

Théorème 4.10. *La compacité est monotone*

Démonstration. Du fait de l'indépendance de la compacité au poids des arêtes externes, elle n'est pas affectée par la diminution du poids des arêtes externes d'une amélioration C-consistante. Nous n'analyserons donc que l'effet de l'augmentation du poids des arêtes à l'intérieur des « clusters ».

$\forall c \in C$, $\forall \pi$ un chemin dans le sous-graphe de G induit par c, $len_G(\pi) \geq len_{G'}(\pi)$ (car tous les poids sont augmentés ou identiques). Ainsi, $diam_G(c) \geq diam_{G'}(c)$. Donc

$$\texttt{Comp}_G(c) = \sum_{(i,j)\in c^2} \frac{w(i,j)}{diam_G(c)} \leq \sum_{(i,j)\in c^2} \frac{w'(i,j)}{diam_{G'}(c)} = \texttt{Comp}_{G'}(c)$$

.

Une amélioration C-consistante implique donc une compacité supérieure ou égale à la compacité initiale. La compacité est monotone. □

Avant de définir la localité, j'ai besoin de définir la notion *d'entente* sur un voisinage pour les graphes.

Définition 4.11 (Entente sur un voisinage)**.** *Soient $G_1 = (V_1, w_1)$ et $G_2 = (V_2, w_2)$ deux graphes et soit $V_a \subseteq V_1 \cap V_2$ un sous-ensemble des noeuds que ces deux graphes ont en commun. Ces graphes* ***s'entendent sur le voisinage*** *de V_a si $w_1(i,j) = w_2(i,j)$ pour tout $i, j \in V_a$ et*

(i) $w_1(i,j) = w_2(i,j)$ *pour tout* $i \in V_a$ *et* $j \in V_1 \cap V_2$,

(ii) $w_1(i,j) = 0$ *pour tout* $i \in V_a$ *et si* $j \in V_1 \setminus V_2$,

(iii) $w_2(i,j) = 0$ *pour tout* $i \in V_a$ *et si* $j \in V_2 \setminus V_1$.

Ceci signifie que les noeuds appartenant à V_a sont connectés exactement aux mêmes noeuds avec les mêmes poids dans les deux graphes.

Définition 4.12 (Localité)**.** *Une fonction de qualité Q est* ***locale*** *si pour tous les graphes $G_1 = (V_1, w_1)$ et $G_2 = (V_2, w_2)$, si $\exists V_a \subseteq V_1 \cap V_2$ tel que G_1 et G_2 s'entendent sur le voisinage de V_a, et pour tout $C_a, D_a \in \mathcal{C}(V_a)$, $C_1 \in \mathcal{C}(V_1 \setminus V_a)$ et $C_2 \in \mathcal{C}(V_2 \setminus V_a)$, si $Q_{G_1}(C_a \cup C_1) \geq Q_{G_1}(D_a \cup C_1)$ alors $Q_{G_2}(C_a \cup C_2) \geq Q_{G_2}(D_a \cup C_2)$.*

Intuition : Un changement local du réseau ne devrait pas avoir d'incidence sur les préférences dans une partie éloignée du réseau. Comme souligné par les auteurs de ces axiomes [95], le problème de la limite de résolution observé par Fortunato et Barthelemy [37] est fortement lié à la satisfaction de la localité. La limite de résolution est le fait que la partition optimale en la fonction de qualité ne peut pas contenir des « clusters » de petite taille par rapport à la taille du graphe.

Théorème 4.13. *La compacité est locale*

Démonstration. Soient $G_1 = (V_1, w_1)$ et $G_2 = (V_2, w_2)$ deux graphes et $V_a \subseteq V_1 \cap V_2$ tel que G_1 et G_2 s'entendent sur le voisinage de V_a. Du fait de la définition 4.11, on a $\forall (i,j) \in V_a^2$, $w_1(i,j) = w_2(i,j)$. Donc $\forall C \in \mathcal{C}(V_a)$, $\texttt{Comp}_{G_1}(C) = \texttt{Comp}_{G_2}(C)$.

On obtient donc $\forall C_a, D_a \in \mathcal{C}(V_a)$, $\forall C_1 \in \mathcal{C}(V_1 \setminus V_a)$ et $\forall C_2 \in \mathcal{C}(V_2 \setminus V_a)$

$$
\begin{aligned}
& \texttt{Comp}_{G_1}(C_a \cup C_1) \geq \texttt{Comp}_{G_1}(D_a \cup C_1) \\
\Leftrightarrow\ & \texttt{Comp}_{G_1}(C_a) + \texttt{Comp}_{G_1}(C_1) \geq \texttt{Comp}_{G_1}(D_a) + \texttt{Comp}_{G_1}(C_1) \\
\Leftrightarrow\ & \texttt{Comp}_{G_1}(C_a) \geq \texttt{Comp}_{G_1}(D_a) \\
\Leftrightarrow\ & \texttt{Comp}_{G_2}(C_a \cup C_2) \geq \texttt{Comp}_{G_2}(D_a \cup C_2)
\end{aligned}
$$

La compacité est donc locale. □

Définition 4.14 (Continuité). *Une fonction de qualité Q est* ***continue*** *si pour tout $\epsilon > 0$ et tout graphe $G = (V, w)$, il existe un $\delta > 0$ tel que pour tout graphe $G' = (V, w')$, si $w(i,j) - \delta < w'(i,j) < w(i,j) + \delta$ pour tous les noeuds i et j, alors $Q_{G'}(C) - \epsilon < Q_G(C) < Q_{G'}(C) + \epsilon$ pour toutes les partitions $C \in \mathcal{C}(V)$.*

Intuition : Une fonction de qualité doit être peu sensible au bruit.

Théorème 4.15. *La compacité est continue*

Démonstration. Cette définition de continuité correspond à la continuité standard d'une fonction à variables multiples. La fonction de distance utilisée est le maximum de la valeur absolue de la différence entre les poids des arêtes entre deux graphes partageant le même ensemble de noeuds. Appelons cette fonction de distance $d((V, w_G), (V, w_{G'})) = \max_{(i,j)\in V^2}(|w_G(i,j) - w_{G'}(i,j)|)$. Notons qu'il s'agit d'une distance au sens formel : les propriétés de positivité, symétrie et d'identité sont immédiates. La preuve de l'inégalité triangulaire suit :

$$\begin{aligned} d(G_x, G_y) + d(G_y, G_z) &= \max_{(i,j)\in V^2}(|w_x(i,j) - w_y(i,j)|) \\ &\quad + \max_{(i,j)\in V^2}(|w_y(i,j) - w_z(i,j)|) \\ &\geq \max_{(i,j)\in V^2}(|w_x(i,j) - w_y(i,j)| + |w_y(i,j) - w_z(i,j)|) \\ &\geq \max_{(i,j)\in V^2}(|w_x(i,j) - w_y(i,j) + w_y(i,j) - w_z(i,j)|) \\ &\geq d(G_x, G_z) \end{aligned}$$

Comme nous considérons une notion de continuité standard, nous pourrons utiliser des propriétés connues, comme la continuité de la combinaison de fonctions continues.

Lemme 4.16. *Pour un graphe connexe $G = (V, w)$, $\forall (a,b) \in V^2$, $dist_G(a, b)$ est continue*

Preuve du lemme 4.16 : Soit $G_n = (V, w_n)$ une séquence de Cauchy de graphes. Alors $\forall (i,j) \in V^2$, $(w_n(i,j))_{n\in\mathbb{N}}$ est aussi une séquence de Cauchy. Par définition, $\exists w$ tel que $w_n(i,j) \to w(i,j)$ (dans ce contexte, $\to$ signifie « converge vers quand $n \to \infty$ ») et il existe un graphe $G = (V, w)$ tel que $G_n \to G$. On suppose G connexe.

$\forall (a, b) \in V^2$, soit $\pi = (a_0 = a, a_1, ..., a_{k-1}, a_k = b)$ un chemin tel que $dist_G(a, b) = len_G(\pi)$, c'est-à-dire un chemin minimal dans G entre a et b.

Si $\exists i \in [0:k-1]$ tel que $w(a_i, a_{i+1}) = 0$, alors $len_G(\pi)$ n'est pas défini et donc ce n'est pas un chemin minimal. Du fait de la définition d'un graphe connexe, un chemin entre a et b existe, donc $\forall i \in [0:k-1]$, $w(a_i, a_{i+1}) > 0$.

Comme $\forall i \in [0:k-1]$, $w_n(a_i, a_{i+1}) \to w(a_i, a_{i+1}) \neq 0$, alors $f(x_0, ..., x_{k-1}) = \sum_{i\in[0:k-1]} \frac{1}{x_i}$ est continue en $(w(a_0, a_1), ..., w(a_{k-1}, a_k))$. Donc $len_{G_n}(\pi) \to len_G(\pi)$.

Comme $dist_{G_n}(a,b) \leq len_{G_n}(\pi)$,

$$\begin{aligned}
\limsup_{n\to+\infty} dist_{G_n}(a,b) &\leq \limsup_{n\to+\infty} len_{G_n}(\pi) \\
&= \lim_{n\to+\infty} len_{G_n}(\pi) = len_G(\pi) = dist_G(a,b) \\
\limsup_{n\to+\infty} dist_{G_n}(a,b) &\leq dist_G(a,b)
\end{aligned}$$

$\liminf_{n\to+\infty} dist_{G_n}(a,b) \leq \limsup_{n\to+\infty} dist_{G_n}(a,b)$, donc

$$\liminf_{n\to+\infty} dist_{G_n}(a,b) \leq \limsup_{n\to+\infty} dist_{G_n}(a,b) \leq dist_G(a,b) \tag{4.2}$$

Soit $\epsilon > 0$. $\limsup_{n\to+\infty} dist_{G_n}(a,b) \leq dist_G(a,b)$, donc $\exists n_0 \in \mathbb{N}$ tel que $\forall n \geq n_0$, $dist_{G_n}(a,b) \leq (1+\epsilon) dist_G(a,b)$.

Soit $n_1 \in \mathbb{N}$ tel que $\forall n \geq n_1$, $d(G, G') \leq \dfrac{1}{2 \times dist_G(a,b)}$.

Par définition, $\forall (i,j) \in V \times V$, $|w_n(i,j) - w(i,j)| \leq \dfrac{1}{2 \times dist_G(a,b)}$.

Pour $n \in \mathbb{N}$, soit π_n un chemin entre a et b tel que $dist_{G_n}(a,b) = len_{G_n}(\pi_n)$. Si π_n n'est pas un chemin dans G pour $n \geq \max(n_0, n_1)$, alors pour $\pi_n = (a_0^{(n)}, ..., a_{k_n}^{(n)})$, $\exists i \in [0 : k_n - 1]$ tel que $w_n(a_i^{(n)}, a_{i+1}^{(n)}) \neq 0$ et $w(a_i^{(n)}, a_{i+1}^{(n)}) = 0$. Dans ce cas,

$$|w_n(a_i^{(n)}, a_{i+1}^{(n)})| = |w_n(a_i^{(n)}, a_{i+1}^{(n)}) - w(a_i^{(n)}, a_{i+1}^{(n)})| \leq \frac{1}{2 \times dist_G(a,b)}$$

$$\Rightarrow len_{G_n}(\pi_n) \geq \frac{1}{w_n(a_i^{(n)}, a_{i+1}^{(n)})} \geq 2 \times dist_G(a,b)$$

Ceci est contradictoire car $len_{G_n}(\pi_n) = dist_{G_n}(a,b) \leq (1+\epsilon) dist_G(a,b)$ pour $n \geq n_0$. Donc, pour $n \geq \max(n_0, n_1)$, π_n est aussi un chemin dans G.

Soit $n_2 \in \mathbb{N}$ tel que $\forall n \geq n_2$, $d(G, G_n) \leq \epsilon w_{min}$ avec $w_{min} = min_{e\in V^2, w(e)\neq 0} w(e)$. D'abord, notons que

$$\forall e \in V^2, w_n(e) \geq w(e) - \epsilon w_{min} \geq (1-\epsilon) w_{min}$$

Ensuite

$$\begin{aligned}
|len_{G_n}(\pi_n) - len_G(\pi_n)| &= |\sum_{i\in[0:k_n-1]} \frac{1}{w_n(a_i^{(n)}, a_{i+1}^{(n)})} - \frac{1}{w(a_i^{(n)}, a_{i+1}^{(n)})}| \\
&= \sum_{i\in[0:k_n-1]} \frac{|w_n(a_i^{(n)}, a_{i+1}^{(n)}) - w(a_i^{(n)}, a_{i+1}^{(n)})|}{w_n(a_i^{(n)}, a_{i+1}^{(n)}) \times w(a_i^{(n)}, a_{i+1}^{(n)})} \\
&\leq \sum_{i\in[0:k_n-1]} \frac{\epsilon w_{min}}{w_n(a_i^{(n)}, a_{i+1}^{(n)}) \times w(a_i^{(n)}, a_{i+1}^{(n)})} \\
&\leq \sum_{i\in[0:k_n-1]} \frac{\epsilon w_{min}}{(1-\epsilon) w_{min} \times w_{min}} \\
|len_{G_n}(\pi_n) - len_G(\pi_n)| &\leq \frac{|V|\epsilon}{w_{min}(1-\epsilon)}
\end{aligned}$$

Du fait que $len_G(\pi_n) \geq dist_G(a,b)$, alors $len_{G_n}(\pi_n) \geq dist_G(a,b) - \frac{|V|\epsilon}{w_{min}(1-\epsilon)}$. Donc $\liminf_{n\to+\infty} len_{G_n}(\pi_n) \geq dist_G(a,b)$. En combinant avec Eq. 4.2 :

$$\limsup_{n\to+\infty} dist_{G_n}(a,b) \leq dist_G(a,b) \leq \liminf_{n\to+\infty} len_{G_n}(\pi_n)$$

$$\Rightarrow \lim_{n\to+\infty} dist_{G_n}(a,b) = dist_G(a,b)$$

Ce qui prouve que la distance entre deux noeuds d'un graphe connexe est une fonction continue.

Fin de la preuve du lemme 4.16

Prouvons maintenant la continuité de la fonction sur des « clusters » non connexes. Afin de simplifier les notations, nous travaillons directement sur les sous-graphes induits en étendant la compacité pour prendre un graphe en entrée :

$$\texttt{Comp}(G) = \begin{cases} 0 & \text{si } |V| = 1 \text{ ou } G \text{ non connexe} \\ \frac{\sum_{e\in V^2} w(e)}{diam(G)} & \text{sinon} \end{cases}$$

Du lemme 4.16, nous savons que $dist(u,v)$ est continue pour tous les graphes connexes, et $dist(u,v) > 0$. Le maximum d'un ensemble de fonctions continues est continu, ce qui signifie que $diam(G)$ est continu pour tous les graphes connexes et $diam(G) > 0$. Une combinaison de fonction continues est continue, et $1/x$ est continue en $\mathbb{R}^+$. $\texttt{Comp}(G)$ est donc continue sur tous les graphes connexes.

Prouvons maintenant que $\texttt{Comp}(G)$ est continue sur des graphes non connexes. De la même manière que dans la preuve du lemme 4.16, nous prenons une

séquence de Cauchy de graphes $(G_n)_{n\in\mathbb{N}} : G_n = (V, w_n) \to G = (V, w)$, mais G est cette fois non connexe. Pour tout $n \in \mathbb{N}$, si G_n est non connexe, $\texttt{Comp}(G_n) = 0 = \texttt{Comp}(G)$.

Si G_n est connexe, soient a et b des noeuds dans deux composantes connexes différentes de G_n. Tout chemin $\pi_n = (a_0^{(n)} = a, a_1^{(n)}, ..., a_k^{(n)} = b) \in \Pi_{G_n}(a, b)$ est tel que $\exists j, w(a_j^{(n)}, a_{j+1}^{(n)}) = 0$. Par définition,

$$len(\pi_n) = \sum_{i\in[0:k-1]} \frac{1}{w_n(a_i^{(n)}, a_{i+1}^{(n)})} \geq \frac{1}{w_{min}^{(n)}}$$

où $w_{min}^{(n)} = \min_{i\in[0:k-1]}(w_n(a_i^{(n)}, a_{i+1}^{(n)}))$. Comme G_n converge vers G, et que a et b sont dans deux composantes connexes différentes de G, $\lim\limits_{n\to+\infty} w_{min}^{(n)} = 0^+$. Donc

$$\lim_{n\to+\infty} dist_{G_n}(u, v) = \lim_{n\to+\infty} \min_{\pi\in\Pi(u,v)} len_{G_n}(\pi) = +\infty$$

.

Comme le diamètre est la distance maximale entre toutes les paires de noeuds, $\lim\limits_{n\to+\infty} diam(G_n) = +\infty$. Par définition d'une séquence de Cauchy, $\lim\limits_{n\to+\infty} \sum_{(i,j)\in V^2} w_n(i,j) = \sum_{(i,j)\in V^2} w(i,j)$. Donc,

$$\lim_{n\to+\infty} \texttt{Comp}(G_n) = \lim_{n\to+\infty} \sum_{e\in V^2} \frac{w_n(e)}{diam(G_n)} = 0 = \texttt{Comp}(G)$$

Ce qui implique que, pour tout graphe non connexe G, `Comp` est continue en G.

Comme la compacité est la somme des $\texttt{Comp}(G)$ appliqués aux sous-graphes induits par la partition, la compacité est continue. □

Dans ce chapitre, j'ai présenté une nouvelle fonction de qualité, la compacité. Celle-ci mesure la vitesse de propagation de la diffusion de l'information dans la communauté dans le pire des cas. Les différents exemples montrent que cette fonction privilégie des structures denses. J'ai aussi prouvé que la compacité satisfait les six axiomes de Van Laarhoven et Marchiori. Ceci implique notamment qu'elle n'est pas affectée par une limite de résolution.

Ce critère de qualité peut être utilisé par un algorithme de détection de communautés. En effet, en simulant des diffusions, il est possible de détecter des communautés. C'est le sujet du prochain chapitre.

Chapitre 5

Une méthode de détection de communautés basée sur l'algorithme du LexDFS

Les algorithmes de parcours de graphe peuvent amener des informations sur la structure du graphe. C'est le cas du LexDFS [23] (« Lexicographical Depth-First Search »), une variante du DFS. Je l'utilise pour un algorithme de détection de communautés, et étudie les communautés ainsi créées.

Les travaux présentés dans ce chapitre ont fait l'objet de deux publications [25, 26].

Weng *et al.* [100] ont observé des phénomènes de diffusions dans twitter. Ils constatent que certaines diffusions se limitent à la communauté dont elles sont originaires. Ces diffusions intra-communautaires ont des caractéristiques proches d'un modèle où, à chaque pas de temps, l'individu ayant le plus de voisins informés devient lui-même informé.

Dans ce modèle, prenons u et v deux voisins tels que u est informé et v ne l'est pas. La probabilité que u transmette l'information à v augmente avec le nombre de voisins de v déjà informés. Il s'agit de la caractéristique du *renforcement*. J'utilise cette caractéristique comme base d'un algorithme afin de détecter les communautés dans lesquelles ces diffusions se concentrent.

Dans ce chapitre, je présente cet algorithme, qui est de complexité log-linéaire. Il est basé sur plusieurs itérations du LexDFS [23], un algorithme de parcours de graphe visitant en priorité les noeuds dont les voisins ont déjà étés visités. J'utilise les parcours produits pour générer des communautés, que j'étudie ensuite expérimentalement.

Zhou et Lipowsky [107] ont montré qu'utiliser des marches aléatoires biaisées pouvait être utilisé pour la détection de communautés. Yucel *et al.* [105] ont récemment proposé d'utiliser des marches aléatoires avec mémoire empêchant le marcheur de revenir sur le dernier noeud visité. Leurs marches sont aussi biaisées de telle manière à augmenter la probabilité de retour sur un noeud déjà visité récemment. Une adaptation de ces techniques permettrait de biaiser des marches aléatoires pour prendre en compte le renforcement. La limite de ces algorithmes est leur complexité. Les auteurs ont montré qu'ils nécessitaient au moins un temps quadratique en la taille du graphe. Mon objectif est d'obtenir un algorithme de complexité raisonnable, utilisable en pratique.

5.1 Un algorithme de partitionnement basé sur le LexDFS

Cette section présente une méthode basée sur l'algorithme de parcours de graphe LexDFS introduit par Corneil et Krueger [23]. Il s'agit d'une variation de l'algorithme DFS (cf Sec. 2.1).

Le LexDFS est de parcourir le graphe visitant en priorité les noeuds dont les voisins ont étés récemment visités. Le LexDFS étiquette chaque noeud avec une liste d'entiers, initialement vide. Lors de la i-ème étape du LexDFS, on récupère tous les voisins non visités du noeud courant. Pour chacun de ces voisins, le nombre i est ajouté au début de son étiquette. Le noeud à visiter à la prochaine étape est choisi uniformément au hasard parmi les noeuds ayant l'étiquette la plus grande dans l'ordre lexicographique.

Un exemple d'exécution du LexDFS est présenté Fig. 5.1. Dans cet exemple, à l'issue de la première étape, le LexDFS choisit entre les trois noeuds ayant l'étiquette [1]. À l'issue de la deuxième étape, le LexDFS ignore le voisin ayant l'étiquette [2] car il n'avait pas été découvert à l'itération précédente.

Le pseudo-code de l'algorithme du LexDFS est présenté Alg. 3. Celui-ci utilise deux attributs des noeuds. L'attribut *lex* est l'étiquette, représentée par un vecteur. L'attribut *visite* indique l'itération à laquelle un noeud a été visité. S'il n'a pas été visité, cet attribut vaut zéro.

J'utilise le LexDFS comme base pour un algorithme de détection de communautés. J'appelle informellement une zone dense un ensembles de noeuds tels que chacun de ces noeuds a un nombre important de voisins à l'intérieur de cet ensemble. Un LexDFS parcourant une zone dense augmente la priorité de visite des autres noeuds de cette zone. Une fois à l'intérieur d'une zone dense, le LexDFS a tendance à en visiter tous les noeuds avant de la quitter.

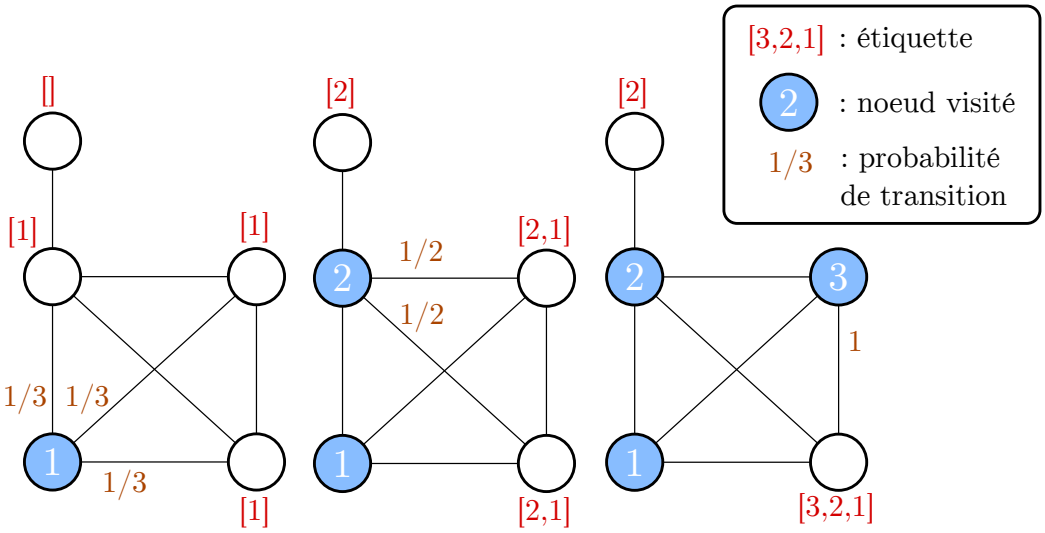


FIGURE 5.1 – Premières étapes d'exécution du LexDFS sur un exemple. Les itérations se lisent de gauche à droite.

J'utilise cette caractéristique pour détecter ces ensembles de noeuds. Nous verrons que cette intuition est confirmée empiriquement.

Je décris maintenant l'algorithme de détection de communautés, le Lex-Clustering. Un score est attribué aux arêtes, présenté Eq. 5.1.

$$\forall (u,v) \in E, score(u,v) = 1 - \frac{|u.visited - v.visited|}{n} \tag{5.1}$$

Plus la visite des deux noeuds incidents à une arête est rapprochée, plus son score est fort. Les arêtes de poids sont de bonnes candidates pour être des arêtes intra-communautaires.

L'algorithme calcule la moyenne de ce score sur un ensemble d'exécutions du LexDFS. Les expériences montrent que 10 d'exécutions suffisent pour s'assurer de la convergence des scores (voir Sec. 5.2). L'algorithme place ensuite chaque noeud dans son propre « cluster ». Enfin, il fusionne successivement les « clusters » reliés par des arêtes à fort score. On peut voir sur l'exemple Fig. 5.2 que des zones denses du graphes sont ainsi révélées. Le pseudo-code associé est présenté Alg. 4

La Fig. 5.2 présente une exécution de la méthode sur un réseau extrait de Facebook, présenté dans la Sec. 5.2. Les arêtes sont affichées si elles sont à l'intérieur d'un « cluster ». Pour des raisons de lisibilité, les « clusters » de taille 1 n'y sont pas représentés. On peut voir dans la Fig. 5.2b qu'une structure dense apparaît. En comparant les Fig. 5.2b et 5.2c, on observe que les « clusters » apparaissent et évoluent séparément avant de fusionner.

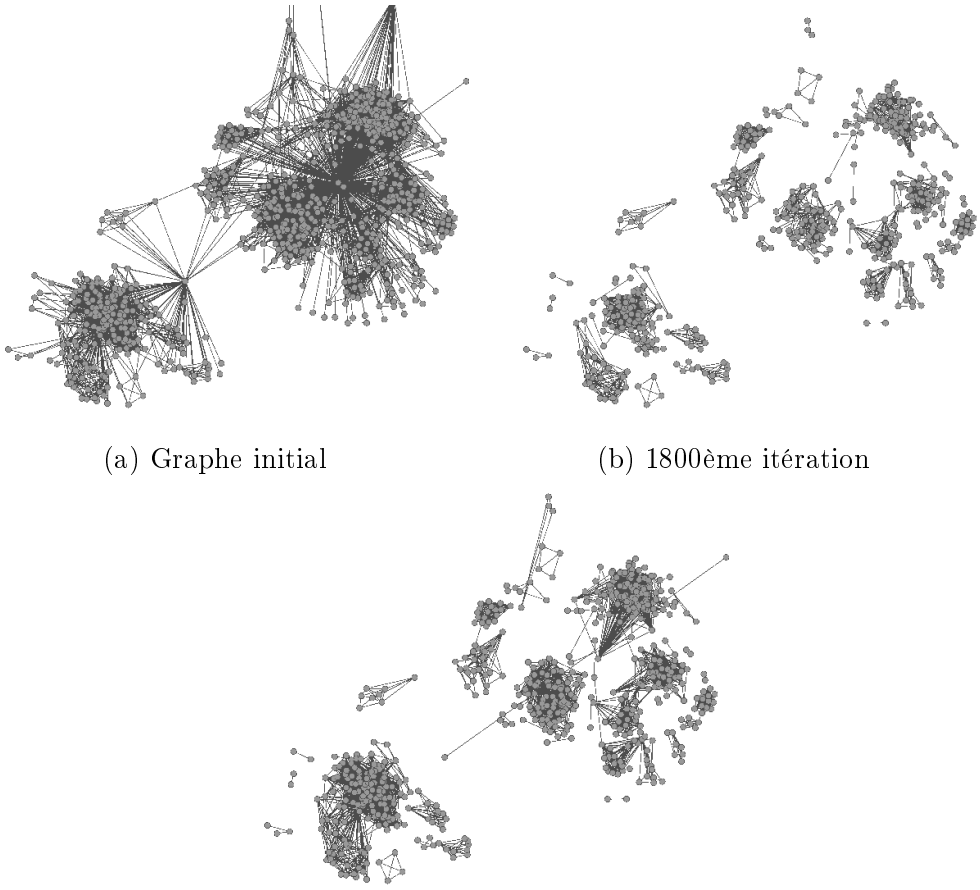
Algorithme 3 : LexDFS(G,s)

```
Données : Un graphe G et un noeud de départ s
début
    // Initialisation des attributs pour tous les noeuds
    pour v ∈ V faire
        v.lex ← ()
        v.visite ← 0
    pile ← (s)
    i ← 1
    tant que pile ≠ () faire
        v ← dépile(pile)
        v.visite ← i                    // Marque le noeud comme visité
        vois ← [ ]
        pour u ∈ voisins(v) faire
            si u.visite = 0 alors
                supprime(pile, u)
                u.lex ← (i, u.lex)          // Ajoute i à l'étiquette
                vois ← [vois, u]
        // Trie par ordre lexicographique des étiquettes.
        // L'ordre des étiquettes égales est aléatoire.
        trie(vois)
        // Place les voisins en haut de la pile
        empile(pile, vois)
        i ← i + 1
```

Notamment, en bas à gauche de la Fig. 5.2c, deux communautés ont fusionné en une seule.

Je montre maintenant que la complexité en temps de la méthode est dans $\mathcal{O}(m \times log(m) + (n + m \log\log(n)) \times l)$, où l est le nombre d'exécutions du LexDFS, m le nombre d'arêtes et n le nombre de noeuds. Il a déjà été montré que le LexDFS est dans $\mathcal{O}(\min(n^2, n + m \log\log(n))$ [22]. Comme je m'intéresse à des graphes dont le nombre d'arêtes est d'ordre inférieur au quadratique [62], la complexité se réduit pour mes applications à $\mathcal{O}(n + m \log\log(n))$.

Je m'intéresse à la complexité de l'Alg. 4. Le calcul des scores des arêtes est dans $\mathcal{O}(m)$, et le tri des arêtes suivant leur score est dans $\mathcal{O}(m \times log(m))$. Les fusions successives de « clusters » est un cas d'union d'ensemble disjoints.



(a) Graphe initial

(b) 1800ème itération

(c) 2700ème itération

FIGURE 5.2 – Les « clusters » d'un extrait du réseau Facebook à différentes étapes du Lex-Clustering

Algorithme 4 : Lex-Clustering(G, l)

Données : Graphe G, nombre d'itérations l

début
  pour $i \leftarrow 1$ **à** l **faire**
    `// Exécute un LexDFS à partir d'un noeud aléatoire`
    LexDFS(G, randnode(V))
    pour $(u,v) \in E$ **faire**
      $s \leftarrow 1 - \frac{|u.visite - v.visite|}{m}$
      $e.score \leftarrow (e.score \times (i-1) + s)/i$
  $ordon \leftarrow tableau(E)$ `// Stocke` E `dans un tableau`
  $trie(ordon)$ `// Trie les arêtes par score décroissant`
  $C \leftarrow \{\{v\} \mid v \in V\}$ `// Crée un cluster par noeud`
  pour $i \leftarrow 1$ **à** $|ordon|$ **faire**
    $(v,u) \leftarrow ordon[i]$
    $fusionne(C, cluster(v), cluster(u))$

Ce problème est résolu par un algorithme quasi-linéaire de Tarjan [90], de complexité $\mathcal{O}(m \times \alpha(m))$ (on effectue au maximum m fusions). La fonction $\alpha(m)$ est l'inverse de la fonction d'Ackerman, qui a une croissance d'ordre supérieur à l'exponentiel. $\alpha(m)$ croît donc plus lentement qu'une fonction log. La complexité est dominée par le terme dans $\mathcal{O}(m \log(m))$.

La complexité finale est dans $\mathcal{O}(m \log(m) + (n + m \log\log(n)) \times l)$.

5.2 Évaluation expérimentale du Lex-Clusering

Cette section présente une étude expérimentale ayant pour but d'analyser les résultats produits par le Lex-Clustering.

Afin de limiter le temps de calcul, j'utilise une heuristique pour approcher le diamètre requis par la compacité (voir Chap. 4). Cette heuristique consiste en deux BFS [67] et s'exécute en temps linéaire. Le premier BFS commence à un noeud aléatoire dans le « cluster ». On calcule ensuite l'excentricité du dernier noeud visité par le premier parcours, en utilisant un second BFS. J'approche le diamètre par cette valeur. Cette heuristique offre une bonne approximation en pratique.

Afin d'analyser le Lex-Clustering, j'utilise les réseaux suivants :

facebook [65] ($n = 4039$, $m = 88234$) est un extrait du réseau social

éponyme. Certains utilisateurs ont installé une application [1] ayant un but explicite de collecte de données pour la recherche. Cette application a collecté la liste d'amis des utilisateurs qui forment ce réseau.

astro [62] ($n = 17903$, $m = 196972$) est un réseau de co-publication scientifiques dans le domaine de l'astrophysique, extrait de la plate-forme de publication en ligne arXiv [2]. Les noeuds sont des auteurs, et il existe une arête entre deux auteurs s'ils ont publié un article ensemble.

enron [55] ($n = 36692$, $m = 183831$) provient de l'entreprise éponyme qui a été forcée par la justice américaine à rendre public l'intégralité des e-mails envoyés et reçus par ses employés [3]. Chaque noeud représente un individu, et il existe une arête entre deux individus si un e-mail a transité de l'un à l'autre.

5.2.1 Vitesse de convergence du Lex-Clustering

J'étudie tout d'abord la vitesse de convergence du Lex-Clustering vers une solution stable. J'utilise la différence de l'ordre des arêtes pour décider de la convergence. En effet, cet ordre détermine directement le comportement de la méthode. Le coefficient de rang de Spearman mesure la corrélation entre deux ordres :

$$S(\mathbf{x}, \mathbf{y}) = 1 - \sum_{i=1}^{|\mathbf{x}|} \frac{6(x_i - y_i)^2}{|\mathbf{x}| \cdot (|\mathbf{x}|^2 - 1)} \tag{5.2}$$

Où $\mathbf{x}$ et $\mathbf{y}$ sont les vecteurs contenant l'ordre des scores. J'observe l'évolution de cette valeur au fur et à mesure des itérations du LexDFS.

Cette expérience est importante pour déterminer la valeur de l dans l'algorithme présenté Alg. 4. La Fig. 5.3 présente les résultats. Les trois jeux de données considérés convergent de la même manière : commençant à 0,5, le coefficient de rang de Spearman dépasse 0,95 après quatre itérations. Il continue ensuite à augmenter et est aux alentours de 0,99 après dix itérations. C'est pour cette raison que j'utilise 10 itérations dans les autres expériences.

1. `http://snap.stanford.edu/socialcircles`
2. `www.arxiv.org`
3. `www.ferc.gov/industries/electric/indus-act/wec/enron/info-release.asp`

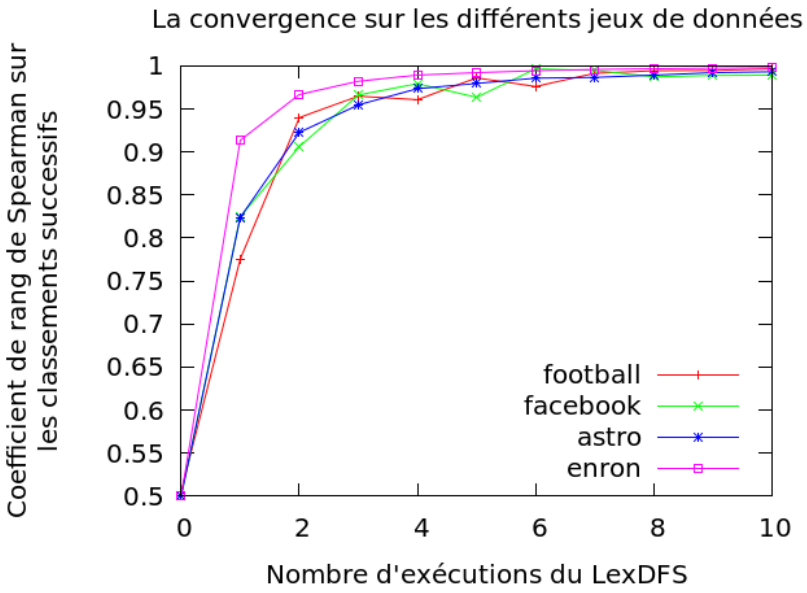


FIGURE 5.3 – Convergence du coefficient de rang de Spearman du score des arêtes sur plusieurs exécutions du LexDFS

5.2.2 Étude des « clusters » individuels

J'analyse maintenant les propriétés des « clusters » que le Lex-Clustering forme au cours du partitionnement hiérarchique. Dans la continuité de la comparaison avec la modularité, je compare les résultats à ceux de l'algorithme de Clauset [20] (voir Sec. 2.4) qui optimise cette valeur de façon gloutonne. Comme il s'agit également d'un algorithme hiérarchique, l'algorithme de Clauset génère donc le même nombre de « clusters » que le Lex-Clustering au cours du processus de fusion. Le nombre de « clusters » observés n'influencera donc pas la comparaison entre les deux algorithmes.

Pour représenter les résultats, j'utilise le NCP plot (« Network Community Profile plot ») introduit par Leskovec *et al.* [63]. Le NCP plot permet de visualiser le résultat d'une mesure par rapport à la taille des « clusters ».

Certaines fonctions de qualité dépendent de la taille de la communauté, comme la modularité et la compacité. Comme les NCP plot représentent déjà la taille en tant que dimension, je préfère utiliser des mesures ne dépendant pas directement de la taille : le diamètre, la conductance et le coefficient de clustering. J'utilise ici la conductance non normalisée [51] décrite Eq. 5.3.

$$q_{cond} = \frac{m(c, V \setminus c)}{Vol(c)} \tag{5.3}$$

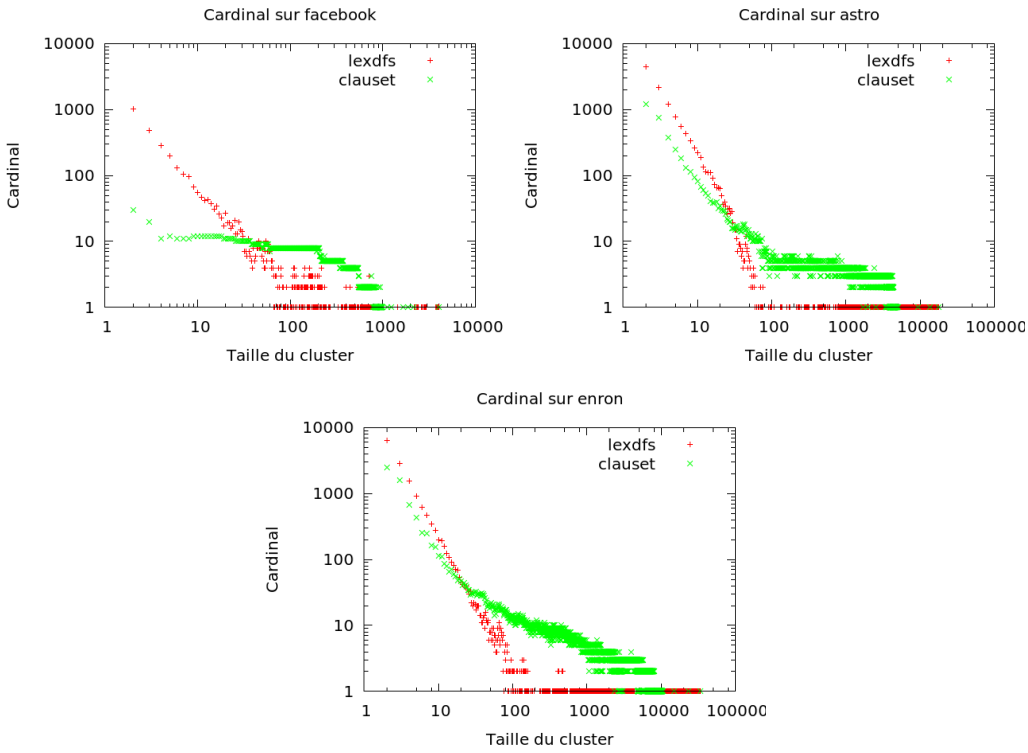


FIGURE 5.4 – Cardinal de l'ensemble des « clusters » de chaque taille

Tout d'abord, je relève Fig. 5.4 la distribution du cardinal des « clusters ». On peut noter que, dans tous les graphes, Lex-Clustering produit beaucoup de « clusters » de taille inférieure à 100. On peut aussi voir qu'il n'existe généralement qu'un « cluster » pour les tailles supérieures à 300. L'algorithme de Clauset, d'un autre côté, produit relativement peu de petits « clusters » et plusieurs grands « clusters ». J'en déduis que le Lex-Clustering produit un « cluster » de taille moyenne pour le fusionner ensuite avec tous les autres.

Les NCP plots sont représentés Fig. 5.5, Fig. 5.6 et Fig. 5.7. Sur chaque graphe, les résultats du Lex-Clustering sont représentés par des croix rouges et ceux de l'algorithme de Clauset en vert. La colonne de gauche de ces figures trace les croix vertes au-dessus des croix rouges, et inversement pour la colonne de droite. Ainsi, la colonne de gauche met les résultats de l'algorithme de Clauset en valeur, tandis que la colonne de droite favorise la visualisation des résultats du Lex-Clustering. Cette représentation est choisie car les résultats se superposent fréquemment.

Les résultats de la conductance sont présentés Fig. 5.5. Les deux algorithmes ont des résultats similaires sur des petits « clusters » de taille inférieure à 80, excepté sur le jeu de données *facebook*, où le Lex-Clustering

produit également des « clusters » de faible conductance. D'un autre côté, les grands « clusters » de taille supérieure à 300 produits par l'algorithme de Clauset ont une conductance plus faible que ceux du Lex-Clustering.

Les NCP plot du coefficient de clustering sont présentés Fig. 5.6. On peut voir sur ces résultats une grande variance du coefficient de clustering pour les deux algorithmes. Les deux algorithmes ne semblent pas optimiser le critère du coefficient de clustering.

Le diamètre des « clusters » individuels est présenté Fig. 5.7. Sur les réseaux *astro* et *enron*, les petits « clusters » du Lex-Clustering ont un diamètre varié, tandis que ceux de Clauset ont un diamètre assez faible. Alors que les grands « clusters » produits par l'algorithme de Clauset présentent un diamètre varié, ceux du Lex-Clustering ont un diamètre relativement faible.

En résumé : on observe que le Lex-Clustering produit un unique « cluster » de taille moyenne qui absorbe tous les autres. Il produit aussi un grand nombre de petits « clusters ». Ces expériences montrent que toutes les fonctions de qualité ne peuvent être utilisées pour évaluer des « clusters ». La multiplicité des définitions des communautés fait que les algorithmes de détection de communautés sont très différents les uns des autres, et c'est aussi le cas pour les métriques. Ces expériences montrent qu'il est difficile pour un algorithme dont la conception de prend pas en compte une métrique de satisfaire celle-ci.

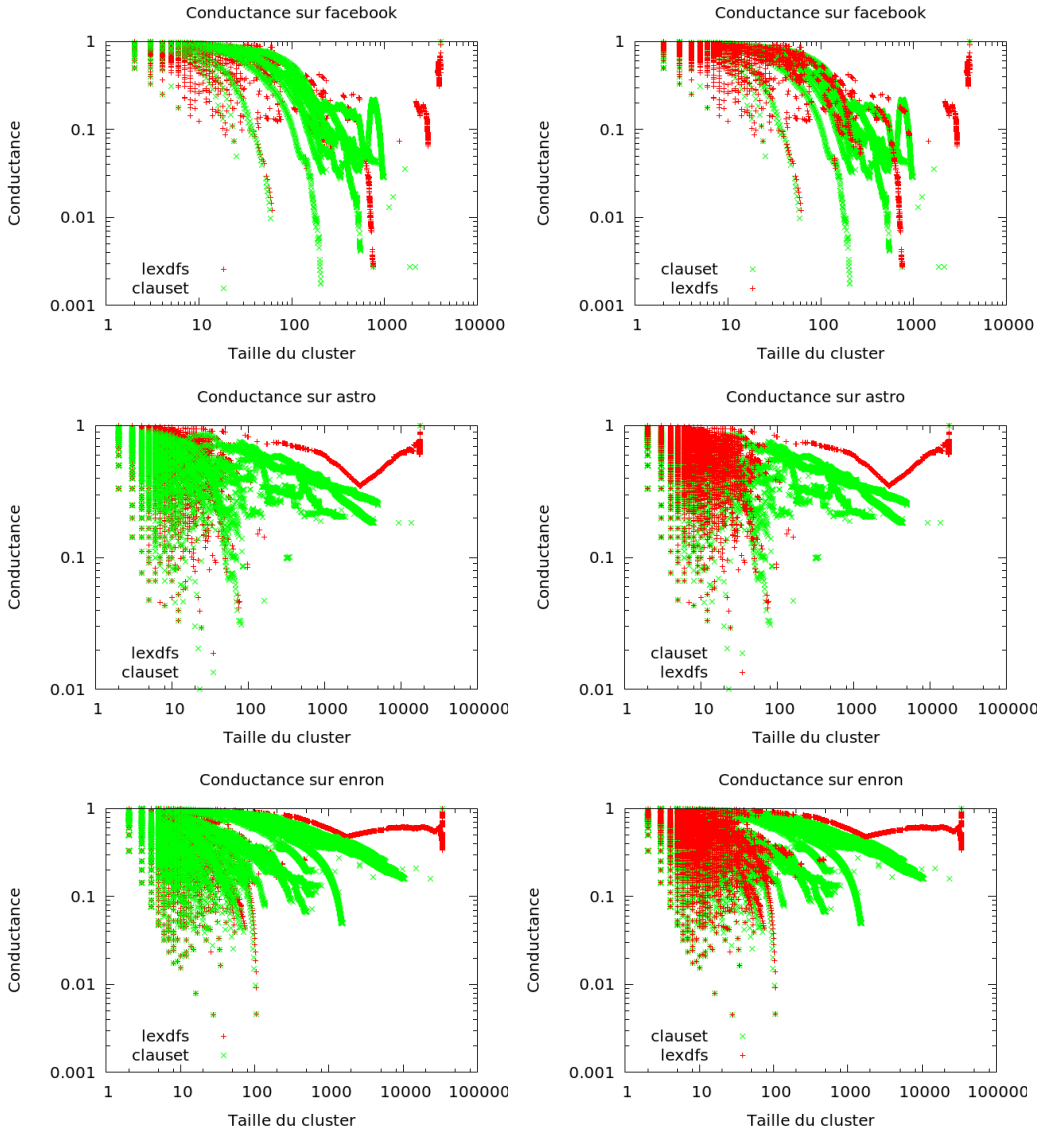


FIGURE 5.5 – NCP plot de la conductance

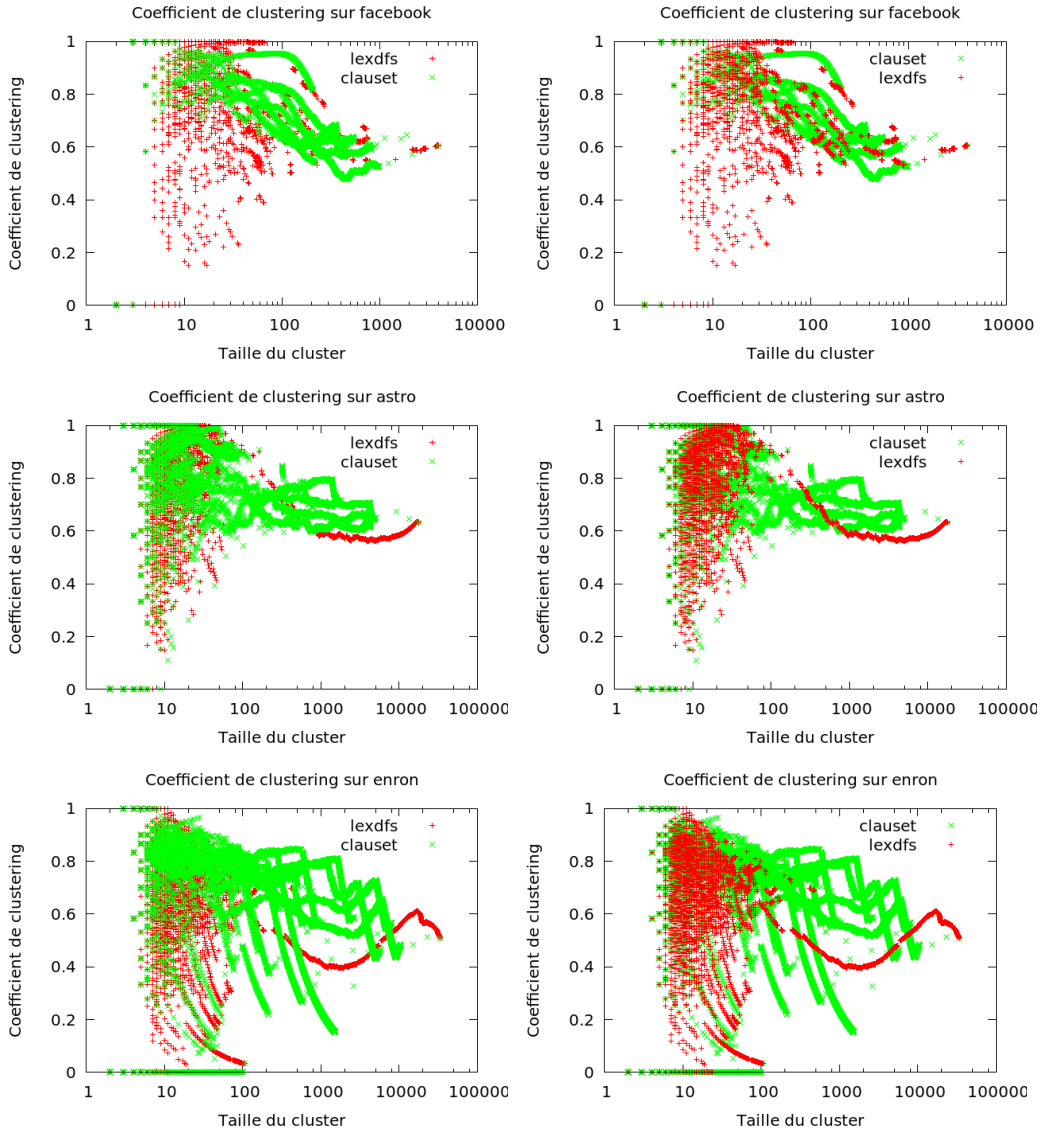


FIGURE 5.6 – NCP plot du coefficient de clustering

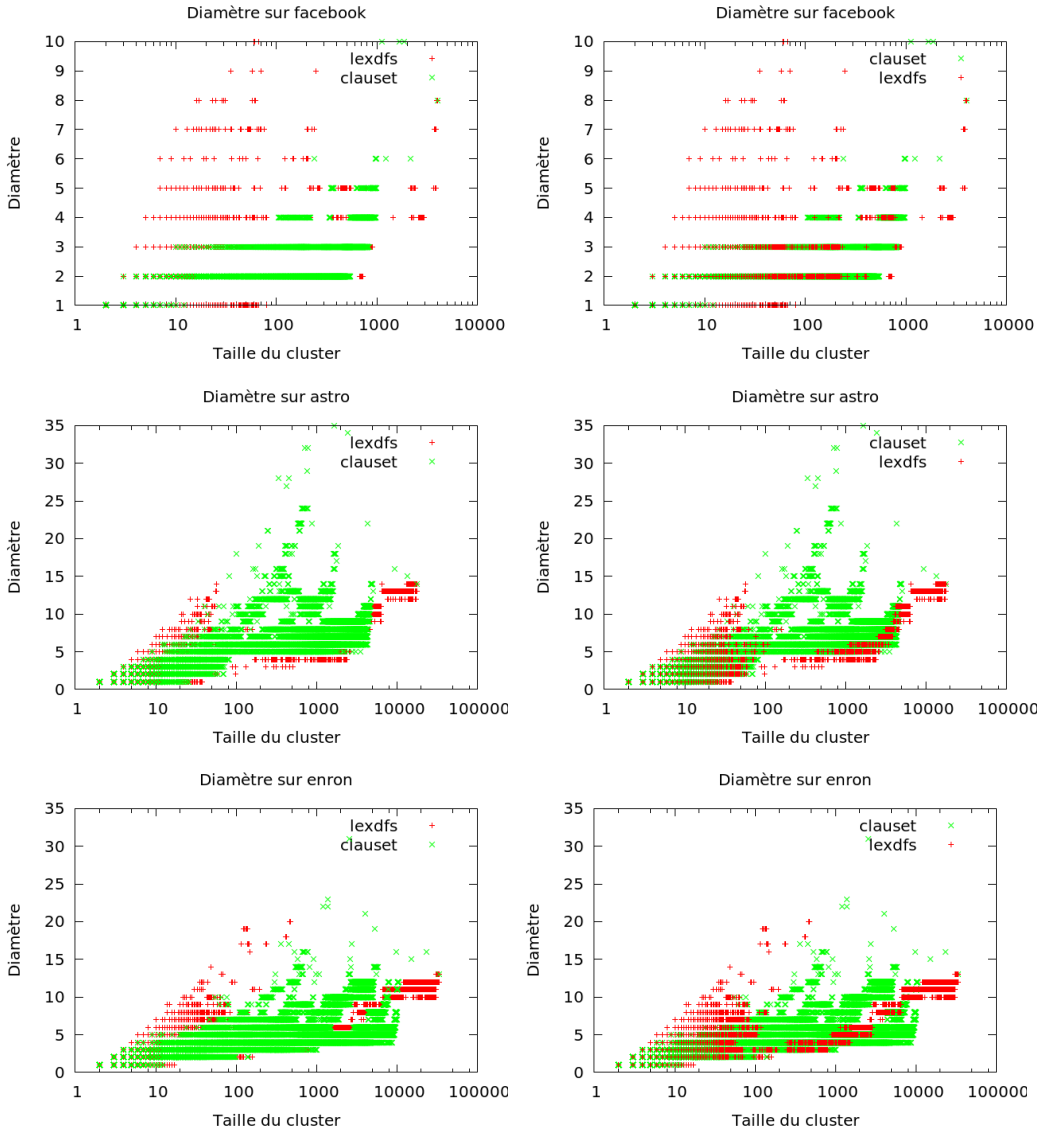


FIGURE 5.7 – NCP plot du diamètre

5.2.3 Étude des partitions

J'analyse maintenant les partitions produites par le Lex-Clustering au cours de son exécution.

Parmi toutes les partitions résultantes d'un algorithme hiérarchique, il est nécessaire d'en sélectionner une. La meilleure partition est traditionnellement [78, 20] sélectionnée en identifiant le maximum d'une fonction de qualité lors du partitionnement hiérarchique. J'appelle **sélecteur** une fonction de qualité dont le maximum global est utilisé pour sélectionner une partition. Dans cette expérience, je compare l'évolution de la modularité, du coefficient de clustering, de la conductance normalisée et de la compacité des partitions produites au cours du partitionnement hiérarchique. L'objectif de cette expérience est donc de déterminer les fonctions qui peuvent être utilisée en tant que sélecteur du Lex-Clustering.

Comme on peut le voir Fig. 5.8 et Fig. 5.9, le coefficient de clustering et la conductance ne semblent pas adaptés pour être des sélecteurs. En effet, on remarque que ce sont des fonctions qui atteignent leur maximum à la fin des fusions, ce que l'on sait être un mauvais partitionnement. Le partitionnement du Lex-Clustering à la fin des fusions de ses « clusters » correspond à un « cluster » géant et à des petits « clusters ». On cherche des petits « clusters » où l'information se diffuse rapidement, ce partitionnement n'est donc pas adapté. Comme vu précédemment, les résultats du Lex-Clustering ne sont pas valorisés par le coefficient de clustering et la conductance.

Comme on peut le voir Fig. 5.10 et Fig. 5.11, la modularité et la compacité sont adaptés pour être utilisé en tant que sélecteurs du Lex-Clustering. En effet, ces deux fonctions produisent un pic clair correspondant à leur maximum global. Ce pic est situé loin de la fin de l'exécution.

Je note aussi que le maximum global ne correspond pas à la même itération suivant la fonction de qualité considérée. Par exemple, la compacité présente un maximum global après 14 000 fusions sur le réseau *enron*. D'un autre côté, la partition correspondant au maximum de conductance se situe aux alentours de 33 000 itérations. Ainsi, le choix du sélecteur peut changer les résultats de manière radicale. J'étudie l'influence du sélecteur dans l'expérience présentée Sec. 5.2.4.

En résumé : la modularité et la compacité sont à-priori de bons sélecteurs. Le coefficient de clustering retourne systématiquement un résultat trivial. La conductance a un bon comportement sur *facebook*, mais ne correspond pas aux attentes sur *enron* et *astro*. Enfin, je remarque que le choix du sélecteur peut avoir une incidence importante sur la partition sélectionnée.

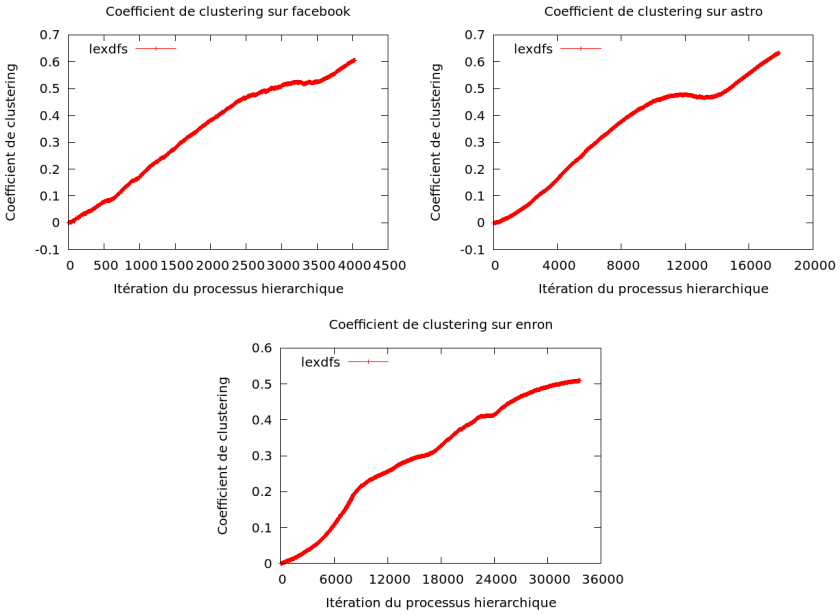


FIGURE 5.8 – Évolution du coefficient de clustering lors des partitionnements hiérarchiques

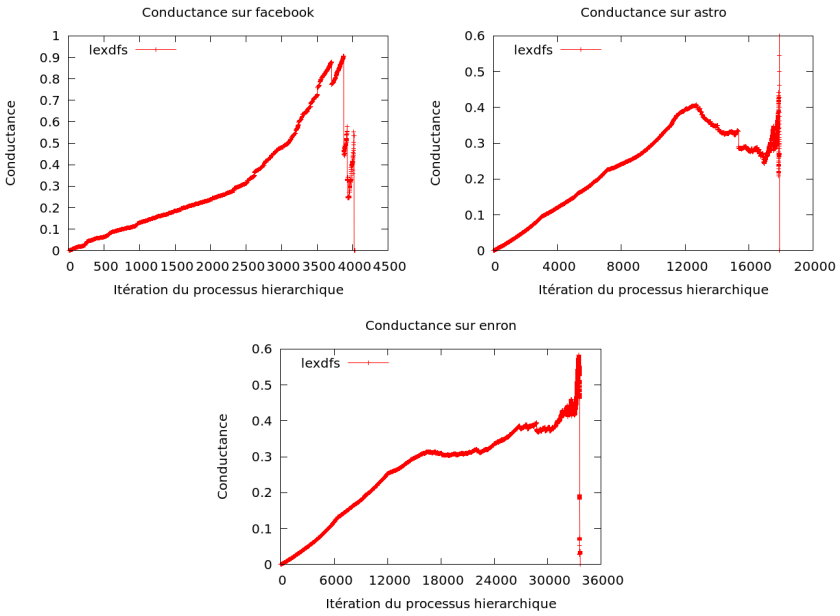


FIGURE 5.9 – Évolution de la conductance lors des partitionnements hiérarchiques

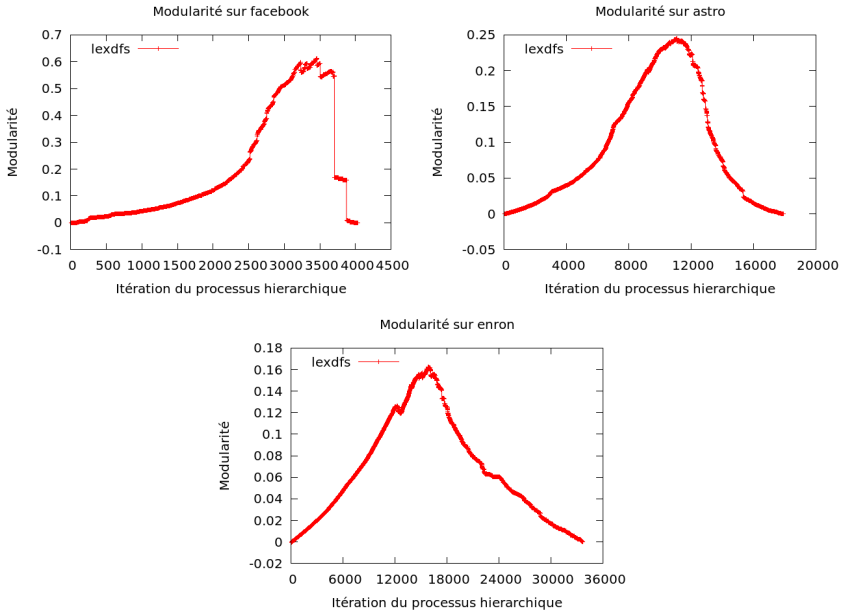


FIGURE 5.10 – Évolution de la modularité lors des partitionnements hiérarchiques

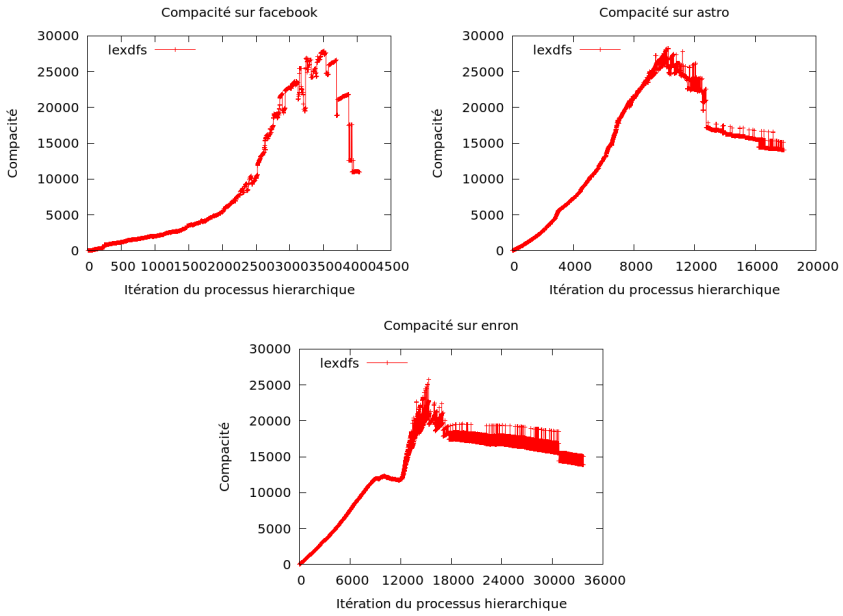


FIGURE 5.11 – Évolution de la compacité lors des partitionnements hiérarchiques

5.2.4 Comparaison avec des vérités de terrain

Je compare ensuite les partitions fournies par différents sélecteurs et des vérités de terrain. Cette expérience mesure si les partitions fournies par le Lex-Clustering ressemblent à celles extraites de l'environnement d'origine du graphe. Les paramètres expérimentaux sont détaillés au Chap. 6, qui présente une méthodologie incluant ces expériences. Celles-ci correspondent à l'étape 3 de la méthodologie présentée Sec. 6.1 de ce chapitre. J'ai exclu le coefficient de clustering en tant que sélecteur dans ces expériences.

	actors	amazon	cora	CS	dblp	football	flickr	github	lj	youtube
$lexdfs_{comp}$	6	2	6	9	2	5	6	4	7	7
$lexdfs_{cond}$	2	3	3	2	4	8	9	2	5	8
$lexdfs_{mod}$	1	4	6	6	3	6	4	1	4	6

TABLE 5.1 – Les rangs fournis par comparaison avec la vérité de terrain (NMI) des résultats du Lex-Clustering avec différents sélecteurs

	actors	amazon	cora	CS	dblp	football	flickr	github	lj	youtube
$lexdfs_{comp}$	3	6	8	6,5	3	3	1	1	1	4
$lexdfs_{cond}$	9	1	2	3	6	8	8	9	9	9
$lexdfs_{mod}$	2	5	6	6,5	2	6	7	3	5	5

TABLE 5.2 – Les rangs fournis par comparaison avec la vérité de terrain (fb3) des résultats du Lex-Clustering avec différents sélecteurs

Pour chaque graphe, j'applique les algorithmes Louvain, Clauset, MCL, Infomap, label propagation et 3-core (cf Sec. 2.4), en plus du Lex-Clustering avec les trois sélecteurs. Le résultat de chacun de ces algorithmes est ensuite comparé aux vérités de terrain de chacun de ces réseaux en utilisant la NMI [58] et la F-BCubed [5]. Les éléments des tables 5.1 et 5.2 présentent les rangs du Lex-Clustering avec différents sélecteurs parmi les algorithmes étudiés. Les algorithmes sont classés en ordre décroissant. Les rangs vont de 1 à 9.

Comme l'expérience précédente le laissait supposer, l'utilisation de différents sélecteurs peut changer radicalement le résultat. Par exemple $lexdfs_{cond}$ est de rang 2 pour cora, F-BCubed tandis que $lexdfs_{comp}$ est de rang 8. On peut voir que la NMI et la F-BCubed fournissent des résultats significativement différents. Le coefficient de rang de Spearman moyen entre les deux classements est de 0,21, ce qui indique en général une corrélation faible.

Chaque rang sera référencé dans le paragraphe suivant sous la forme *rang-nmi/rang-fbcubed*.

On remarque tout d'abord que la conductance fournissent des partitions relativement proches de celles d'*amazon* par rapport aux autres algorithmes : 3/1. La conductance comme sélecteur du Lex-Clustering produit de très bons scores sur ces graphes *amazon* (3/1), *cora* (3/2) et *CS* (4/2). Les partitions sélectionnées par cette fonction de qualité ont cependant de très mauvais résultats sur le reste des jeux de données. La compacité et la modularité ont des résultats relativement similaires : de bons scores sur les données *dblp* (2/3, 3/2), *actors* (6/3, 1/2) et *github* (4/1, 1/3).

En résumé, comparer de multiples fonctions de qualité en tant que sélecteurs du Lex-Clustering permet d'obtenir des résultats significativement différents. La conductance a de très bons résultats sur les réseaux liés à la publication scientifique (un contexte, comme décrit Chap. 6). L'utilisation de la compacité et de la modularité donne de bons résultats sur des réseaux de collaboration : *dblp*, *actors* et *github*. La qualité d'un sélecteur est donc dépendante de la vérité de terrain. On remarque que certains sélecteurs correspondent à plusieurs vérités de terrain.

En m'inspirant d'études portant sur la diffusion de l'information dans les réseaux sociaux, j'ai proposé d'utiliser le LexDFS pour détecter des communautés. Cet algorithme de parcours de graphe visite les zones denses du graphe en un nombre faible d'itérations. J'ai utilisé cette propriété dans un algorithme hiérarchique de complexité log-linéaire, le Lex-Clustering. Les expériences ont montré que cet algorithme produit des résultats très proches de ce qui est observé dans différents jeux de données, suivant la fonction de qualité utilisée comme sélecteur.

Nous avons vu que l'on pouvait évaluer un algorithme de détection de communautés de plusieurs façons. Des fonctions de qualité et des jeux de données peuvent être utilisés pour cette évaluation. D'un côté, les propriétés mesurées par les fonctions de qualité sont bien identifiées. D'un autre côté, les implications de la proximité des résultats avec les vérités de terrain sont moins claires. Dans le prochain chapitre, j'étudie la relation entre les deux méthodes d'évaluation.

Chapitre 6

L'évaluation de structures communautaires

On peut évaluer les partitions d'un graphe en appliquant des fonctions de qualité et en les comparant à des vérités de terrain. Dans ce chapitre, je fais le lien entre les deux méthodes en analysant la corrélation entre les classements des partitions selon chaque évaluation.

Les travaux présentés dans ce chapitre ont fait l'objet de la publication [27].

Yang et Leskovec [103] distinguent communautés structurelles et fonctionnelles. Les communautés structurelles sont définies par les propriétés du graphe, tandis que les communautés fonctionnelles sont caractérisées par un passé ou un intérêt commun de leurs membres.

Ces communautés fonctionnelles apparaissent notamment dans les métadonnées associées aux réseaux. Par exemple, on trouve l'appartenance explicite à des groupes dans les métadonnées des sites web de réseaux sociaux.

Une motivation de la détection de communautés peut être la recherche de communautés fonctionnelles. Dans ce but, les algorithmes sont comparés aux métadonnées, utilisées alors comme vérité de terrain (par exemple [52, 79]). Un algorithme est considéré comme performant dans ce cas si les partitions qu'il produit sont similaires aux métadonnées.

Je définis la cohérence entre fonctions de qualité et vérités de terrain. Soit une fonction de qualité Q. Soit une vérité de terrain M d'un graphe G. Soit un ensemble A de partitions de G. En utilisant Q pour évaluer les partitions de A, on obtient un classement $C_Q(A)$ des partitions. De la même manière, on obtient un classement $C_M(A)$ en comparant avec la vérité de terrain. Je qualifie la fonction de qualité Q et la vérité de terrain M de *cohérentes* si

$C_Q(A)$ et $C_M(A)$ ont un coefficient de rang de Spearman important (Eq. 5.2).

Dans ce chapitre, j'analyse la cohérence entre fonctions de qualité et vérités de terrain. J'identifie des vérités de terrain ayant une cohérence similaire avec les fonctions de qualité, et j'appelle ces ensembles des *contextes*. Je détermine aussi les fonctions de qualité correspondant le mieux à chaque contexte.

Utiliser ces vérités de terrain a récemment été critiqué par Peel *et al.* [77]. Ils présentent quatre raisons pour que le résultat d'un algorithme de détection de communautés ne soit pas similaire à la vérité de terrain :

1. la structure de la vérité de terrain n'est pas liée à la structure du réseau,
2. la structure de la vérité de terrain et l'algorithme de détection de communautés capturent des aspects différents du réseau,
3. le réseau ne présente pas de structure communautaire,
4. l'algorithme de détection de communautés n'est pas adapté à la détection des communautés fonctionnelles.

Pour les auteurs, utiliser les métadonnées comme vérité de terrain équivaut à supposer que seul le cas 4 est possible.

Tout d'abord, il n'est pas clair que les cas 2 et 4 sont différents. En effet, chaque algorithme capture un aspect différent du réseau, et c'est la pertinence de cet aspect qui fait la pertinence de l'algorithme. Ensuite, les cas 1 et 3 impliquent que le réseau et les métadonnées ne correspondent pas aux suppositions que je fais sur les propriétés des réseaux sociaux. Je considère donc la comparaison des partitions avec les vérités de terrain comme pertinente pour la recherche de communautés fonctionnelles.

Le lien entre les communautés fonctionnelles et structurelles est aussi étudié par Hric *et al.* [49]. Ils montrent que les algorithmes de détection de communautés actuels produisent des partitions assez éloignées des vérités de terrain et soulignent la nécessité d'algorithmes plus adaptés.

Les fonctions de qualité ont étés étudiées par d'autres auteurs.

Yang et Leskovec [103] ont analysé 12 fonctions de qualité pouvant être appliquées au niveau de la communauté. Leurs expériences identifient les propriétés que mesurent ces fonctions de qualité.

Almeida *et al.* [2] ont comparé le résultat de quatre algorithmes en utilisant cinq fonctions de qualité sur cinq jeux de données. Ils ont notamment observé que certaines fonctions de qualité ont tendance à favoriser des « clusters » de plus grande taille, et inversement.

Chakraborty *et al.* [18] ont utilisé une méthodologie similaire à celle que je mets en place dans le but de mesurer la performance de leur fonction de qualité sur quelques jeux de données. Les approches diffèrent donc par le but, mais aussi par l'échelle : les expériences que je mets en place incluent un nombre bien plus important de jeux de données, fonctions de qualité et d'algorithmes.

6.1 Méthodologie expérimentale

Mes expériences ont deux objectifs. Tout d'abord, j'identifie des contextes. Ensuite, je repère les fonctions de qualité cohérentes avec chaque contexte.

Pour chaque graphe ayant une vérité de terrain, j'exécute les étapes suivantes :

1. L'application d'algorithmes de détection de communautés sur le graphe.
2. Le calcul de fonctions de qualité sur les partitions résultantes.
3. La comparaison des partitions de l'étape 1 avec la vérité de terrain, créant un score de référence pour chaque partition.
4. La mesure de la cohérence entre chaque vérité de terrain et chaque fonction de qualité.

Les étapes de cette méthodologie sont représentées Fig. 6.1.

Après l'exécution de ces étapes pour chaque jeu de données, j'applique le coefficient de rang de Spearman entre les résultats de cohérence des différentes fonctions de qualité pour chaque paire de graphes. Cette dernière étape permet l'identification de contextes. Je peux ensuite, par l'analyse des mesures produite par l'étape 4, reconnaître les fonctions de qualité cohérentes avec des contextes.

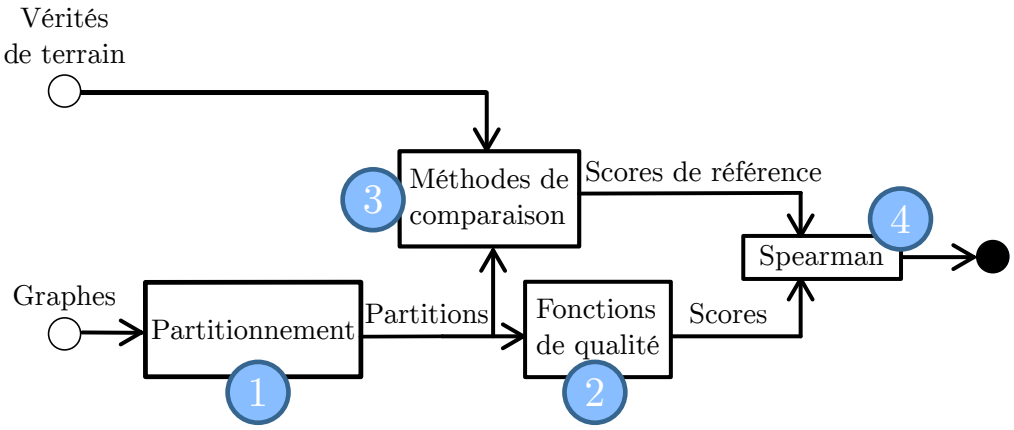


FIGURE 6.1 – Étapes de la méthodologie expérimentale

6.2 Paramètres expérimentaux

Les algorithmes de détection de communauté utilisés sont Louvain, Clauset, MCL, Infomap, Label propagation et 3-core (voir Sec. 2.4) ainsi que LexDFS (voir Chap. 5). Les fonctions de qualité utilisées sont Coefficient de clustering, Permanence, 1-Flake-ODF, FOMD, 1-Cut-ratio, 1-Conductance, Modularité, Surprise et Signifiance (voir Sec. 2.3) ainsi que la Compacité (voir Chap. 4) Les méthodes de comparaison utilisées sont la NMI et la fb3. Elles sont décrites Sec. 2.5.

J'applique la méthodologie sur 10 réseaux. Les noeuds qui n'appartiennent à aucun « cluster » dans les vérités de terrain sont supprimés et seule la composante connexe la plus grande est utilisée. Je note $\rightarrow$ pour les réseaux orientés, et ∞ pour les communautés chevauchantes.

Réseaux de collaboration (voir Tab. 6.1). Ces réseaux représentent des individus travaillant ensemble.

Le réseau « Computer Science » (CS) provient de la même source que le réseau « DBLP ». CS est cependant composé uniquement de scientifiques travaillant en informatique. De plus, les vérités de terrain sont différentes. Les réseaux « actors » et « github » sont construit à partir de graphes bipartis, et sont composés de cliques à l'intérieur des communautés.

1. `http://snap.stanford.edu/data/`
2. `http://konect.uni-koblenz.de`
3. `https://github.com/blog/466-the-2009-github-contest`

Nom	n	m	noeuds	arêtes	communautés
DBLP[1] [103]	130k	333k	auteurs	co-auteur	publié dans le même ouvrage ⊚
CS [16][17]	401k	1428k	auteurs	co-auteur	domaine de publication ⊚
Actors (imdb)[2] [8]	124k	20490k	acteurs	film commun	films ⊚
Github[23]	40k	22278k	développeurs	co-contributions	projets ⊚

TABLE 6.1 – Réseaux de collaboration

Nom	n	m	noeuds	arêtes	communautés
LiveJournal[1] [103]	1143k	16881k	bloggeurs	« follow » →	groupes explicites ⊚
Youtube[1] [103]	51k	317k	youtubeurs	« follow » →	groupes explicites ⊚
Flickr [71]	368k	11916k	utilisateurs	« follow » →	groupes explicites ⊚

TABLE 6.2 – Sites web de réseaux sociaux

Sites web de réseaux sociaux (« Online Social Networks », OSN) (voir Tab. 6.2). Ces réseaux représentent des relations explicites entre individus sur des sites web.

Le site web « LiveJournal » permet de poster des blogs en ligne. « Youtube » est une plate-forme de partage de vidéos. L'activité principale sur « Flickr » est le partage de photographies. Ces sites incluent la possibilité de se connecter entre utilisateurs. Ils permettent aux utilisateurs de rejoindre librement des groupes explicites.

Sur ces sites, les liens sont orientés. Les auteurs originaux ont cependant considéré que les traiter comme des réseaux non dirigés était raisonnable du fait que la plupart des liens existaient dans les deux sens.

Réseaux proches des réseaux sociaux (voir Tab. 6.3). Les noeuds de ces réseaux ne représentent pas des individus, mais les réseaux en eux-mêmes décrivent une forme d'interaction sociale.

1. `http://snap.stanford.edu/data/`

Nom	n	m	noeuds	arêtes	communautés
Amazon[1] [103]	148k	267k	produits	achat commun fréquent	catégories
Football [39]	115	613	équipes de football	1 match ou plus	divisions
Cora[2] [88]	23k	89k	articles scientifiques	citations →	catégories

TABLE 6.3 – Réseaux semi-sociaux

« Amazon » renseigne les produits fréquemment achetés avec ceux que l'on consulte. Les catégories de produits forment la vérité de terrain. Le réseau « Football » correspond aux matchs d'équipes de football universitaires américaines en automne 2000, et les métadonnées sont les divisions. Le réseau « Cora » est formé d'un ensemble de publications scientifiques provenant d'un dépôt en ligne. Les publications sont liées par une arête si l'une cite l'autre, et les métadonnées sont les domaines scientifiques des papiers.

Réseaux artificiels J'utilise le benchmark de Lancichinetti-Fortunato-Radicchi (LFR) [59] comme outil d'étude de la méthodologie. Il s'agit d'un modèle de génération de graphes et de structures communautaires.

Plusieurs paramètres permettent de modifier le comportement de ce modèle :

— n : le nombre de noeuds,

— $\hat{k}$: le degré moyen,

— k_{max} : le degré maximum,

— μ : le ratio degré externe/degré interne des noeuds,

— t_1 : le coefficient de la loi de puissance de la distribution des degrés

— t_2 : le coefficient de la loi de puissance de la distribution des tailles des communautés,

— $\hat{cc}$: coefficient de clustering moyen,

— on : le nombre de noeuds appartenant à plusieurs communautés,

— om : le nombre de communautés auxquels ceux-ci appartiennent.

1. `http://snap.stanford.edu/data/`
2. `http://konect.uni-koblenz.de`

nom	n	$\hat{k}$	k_{max}	μ	t_1	t_2	$\hat{cc}$	on	om
LFRa	10 000	50	1k	0.1	2.5	2.5	0.2	8k	4
LFRb	100 000	50	2,5k	0.1	2.5	2.5	0.2	80k	4
LFRc	10 000	100	500	0.4	2.1	2.0	0.1	8k	5
LFRd	10 000	50	1k	0.1	2.5	2.5	0.2	0	0
LFRe	10 000	100	500	0.4	2.1	2.0	0.1	0	0

TABLE 6.4 – Les paramètres des cinq classes de réseaux artificiels LFR

Comme présenté à la Tab. 6.4, je compose 5 classes de réseaux. Celles-ci sont différenciées par les paramètres du LFR qui les génèrent. Je les crée de telle manière que les classes a, b et d ont beaucoup de paramètres du LFR en commun, et les classes c et e sont différentes. Les réseaux de la classe a présentent des communautés séparées (μ faible) et chevauchantes. La classe b a les mêmes paramètres que la classe a, mais avec 10 fois plus de noeuds. Notons que le degré maximal n'évolue pas de manière linéaire avec l'augmentation de taille, mais par la loi de puissance de paramètre 2,5. Ainsi une multiplication par 10 entraîne une augmentation par $10^{1/2,5} \sim 2{,}5$.

La classe c a des paramètres très différents de la classe a, sauf en ce qui concerne la taille. Les classes d et e sont respectivement similaires aux classes a et c, mais les communautés n'y sont pas chevauchantes.

6.3 Optimisations

Trois mesures prennent beaucoup de ressources dans ces expériences : le calcul du diamètre, le comptage des triangles et la fb3. Pour le calcul du diamètre, j'utilise la même approche que celle développée Sec. 5.2. f_{clus} et f_{perm} nécessitent le calcul du nombre de triangles internes qui peut être très grand pour des graphes ayant des zones très denses. La fb3 est en $\mathcal{O}(|c|^2)$ en temps pour le calcul des valeurs de la communauté c, ce qui pose problème pour les communautés de taille importante.

J'échantillonne ces valeurs. La borne de Hoeffding [46] permet de borner l'erreur ainsi faite, les échantillons étant $i.i.d$ et dans l'intervalle [0,1]. Cette borne donne une indication du nombre d'échantillons t nécessaires pour obtenir une précision satisfaisante. Soit p la probabilité que l'erreur résultant de l'échantillonnage dépasse ϵ. La borne est alors :

$$P(|\overline{X} - E[X]| \geq \epsilon) = p \leq 2e^{-2n\epsilon^2} \Leftrightarrow n \geq \frac{ln(p/2)}{-2\epsilon^2} \tag{6.1}$$

J'utilise 5000 échantillons, ce que signifie que $p \leq 5\%$ et $\epsilon \leq 0.02$. En pra-

tique, on observe des erreurs d'environ 10^{-4}, ce qui est trop faible pour perturber les résultats.

Du fait que je prenne en compte le classement des mesures, des erreurs bornées peuvent avoir des effets non bornés si des mesures ont des résultats très proches. De plus, quelques algorithmes de détection de communauté sont aléatoires, ce qui implique une potentielle différence incontrôlable dans les résultats. Afin d'observer l'influence de ces effets, les calculs ont étés effectués entièrement plusieurs fois. Il n'a été observé que des différences minimes entre les résultats.

6.4 Résultats

J'applique la méthodologie sur les graphes générés par le LFR. Pour chaque classe, je génère aléatoirement trois réseaux avec les paramètres décrits Sec. 6.2. Ainsi, je peux observer l'effet de l'aléatoire dans le processus de génération du LFR sur la stabilité de la méthode. Les résultats sont reportés Tab. 6.5 et Tab. 6.6 pour la NMI et la fb3 respectivement.

	a1	a2	a3	b1	b2	b3	c1	c2	c3	d1	d2	d3	e1	e2	e3
a1	-	1.00	1.00	0.98	0.99	0.45	0.40	-0.23	-0.09	0.53	0.53	0.53	0.69	0.31	0.95
a2	-	-	1.00	0.98	0.99	0.45	0.40	-0.23	-0.09	0.53	0.53	0.53	0.69	0.31	0.95
a3	-	-	-	0.98	0.99	0.45	0.43	-0.22	-0.07	0.53	0.53	0.53	0.69	0.31	0.94
b1	-	-	-	-	0.99	0.48	0.35	-0.15	0.01	0.53	0.53	0.53	0.69	0.31	0.92
b2	-	-	-	-	-	0.46	0.34	-0.21	-0.07	0.53	0.53	0.53	0.69	0.31	0.94
b3	-	-	-	-	-	-	0.29	0.03	0.21	0.15	0.15	0.15	0.67	0.56	0.42
c1	-	-	-	-	-	-	-	0.07	0.15	0.27	0.27	0.27	0.30	0.25	0.41
c2	-	-	-	-	-	-	-	-	0.90	0.34	0.34	0.34	-0.12	0.05	-0.32
c3	-	-	-	-	-	-	-	-	-	0.22	0.22	0.22	-0.03	0.20	-0.16
d1	-	-	-	-	-	-	-	-	-	-	1.00	1.00	0.47	0.12	0.47
d2	-	-	-	-	-	-	-	-	-	-	-	1.00	0.47	0.12	0.47
d3	-	-	-	-	-	-	-	-	-	-	-	-	0.47	0.12	0.47
e1	-	-	-	-	-	-	-	-	-	-	-	-	-	0.76	0.70
e2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.35
e3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

TABLE 6.5 – La corrélation entre le classement des fonctions de qualité fournies par la NMI pour les graphes synthétiques du LFR

Dans la Tab. 6.5, on peut voir que les scores intra-classes sont particulièrement élevés. Les fonctions de qualité ont donc des classements similaires pour les graphes des mêmes classes. Il y a cependant quelques exceptions.

Par exemple, le classement des fonctions de qualité sur c1 semble être plus proche de celui des graphes de la classe *a* que de sa propre classe. L'existence de ces variations est due à la nature aléatoire du modèle générateur. La méthodologie utilisant la NMI est donc sensible à des variations dues aux paramètres non contrôlés par le modèle LFR.

Les similarités attendues sont observées : les graphes de la classe *a* ont des scores globalement proches de ceux des classes *b* et *d*. Cependant, les graphes de la classe *e* sont proches de ceux de la classe *a*, une observation qui est aussi vraie pour la fb3 (cf Tab. 6.6). Le fait que les communautés de la classe *e* ne soient pas chevauchantes semble donc diminuer l'impact de la différence des autres paramètres sur la proximité entre les classes *a* et *e*.

Ainsi, l'application de la méthodologie avec la NMI comme méthode de comparaison a globalement le comportement attendu. La procédure est parfois influencée par des caractéristiques aléatoires dans le modèle LFR.

	a1	a2	a3	b1	b2	b3	c1	c2	c3	d1	d2	d3	e1	e2	e3
a1	-	0.99	1.00	0.95	0.94	0.96	-0.23	-0.50	-0.44	0.72	0.72	0.72	0.89	0.89	0.88
a2	-	-	0.99	0.94	0.93	0.94	-0.26	-0.48	-0.42	0.70	0.70	0.70	0.89	0.87	0.88
a3	-	-	-	0.95	0.94	0.96	-0.23	-0.50	-0.44	0.72	0.72	0.72	0.89	0.89	0.88
b1	-	-	-	-	1.00	1.00	-0.26	-0.49	-0.44	0.80	0.80	0.80	0.90	0.92	0.93
b2	-	-	-	-	-	1.00	-0.27	-0.48	-0.43	0.80	0.80	0.80	0.90	0.92	0.93
b3	-	-	-	-	-	-	-0.25	-0.49	-0.43	0.80	0.80	0.80	0.90	0.91	0.92
c1	-	-	-	-	-	-	-	0.62	0.76	-0.27	-0.27	-0.27	-0.42	-0.38	-0.37
c2	-	-	-	-	-	-	-	-	0.90	-0.26	-0.26	-0.26	-0.49	-0.51	-0.37
c3	-	-	-	-	-	-	-	-	-	-0.31	-0.31	-0.31	-0.47	-0.47	-0.39
d1	-	-	-	-	-	-	-	-	-	-	1.00	1.00	0.75	0.78	0.79
d2	-	-	-	-	-	-	-	-	-	-	-	1.00	0.75	0.78	0.79
d3	-	-	-	-	-	-	-	-	-	-	-	-	0.75	0.78	0.79
e1	-	-	-	-	-	-	-	-	-	-	-	-	-	0.98	0.96
e2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.96
e3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

TABLE 6.6 – La corrélation entre le classement des fonctions de qualité fournies par la NMI pour les graphes synthétiques du LFR

Dans la Tab. 6.6, on observe que les corrélations en utilisant la fb3 ne présente aucune valeur entre -0.2 et 0.6. Il n'y a donc pas de corrélation faible. La classe *c* est considérée comme très différente des autres car ses résultats sont tous anticorrélés avec ceux des autres classes. La corrélation intra-classes est particulièrement forte, supérieure à 0.95 pour toutes les classes sauf la *c* où l'aléatoire semble avoir un léger rôle, avec une corrélation c1/c2 à 0.62.

Comme indiqué précédemment, les résultats de la fb3 ne semblent pas prendre en compte la différence de paramètres de la classe e.

En conclusion, les résultats de la fb3 sont moins sensibles à la génération aléatoire du LFR que ceux de la NMI. Cependant, la différence de recouvrement fait que la classe e est considérée comme ressemblante aux autres.

J'applique la même méthodologie sur l'ensemble des données rassemblées. L'objectif est alors d'identifier les *contextes*. Contrairement au LFR, il n'y a pas de contrôle précis des propriétés des réseaux en question, simplement la connaissance de l'environnement d'origine des réseaux.

Les résultats des jeux de données réels sont bien moins extrêmes que ceux des graphes générés. Néanmoins, on peut voir Tab. 6.7 et 6.8 que les tuples de jeux de données **(cora, CS)** et **(flickr, lj, youtube)** ont un score important par les deux méthodes de comparaison.

Cette observation est explicable par la similarité de l'origine de ces réseaux. Même si ce que représentent les noeuds est différent, Cora et CS correspondent tous deux à des publications scientifiques et leurs vérités de terrain à des domaines de publication. Youtube, flickr et lj ont aussi des mécaniques similaires : la création de lien est due à un individu suivant les publications d'un autre, et les structures communautaires sont des groupes explicites. Les autres corrélations observées diffèrent suivant la méthode de comparaison utilisée.

	CS	actors	amazon	cora	dblp	flickr	football	github	lj	youtube
CS	-	0.923	0.281	0.972	0.302	0.103	0.245	0.014	0.253	-0.187
actors	-	-	0.264	0.899	0.276	0.168	0.318	0.105	0.262	-0.077
amazon	-	-	-	0.280	0.965	-0.231	0.523	-0.033	-0.269	-0.336
cora	-	-	-	-	0.327	0.115	0.213	0.052	0.334	-0.191
dblp	-	-	-	-	-	-0.238	0.453	0.031	-0.241	-0.357
flickr	-	-	-	-	-	-	0.180	0.367	0.808	0.759
football	-	-	-	-	-	-	-	0.350	-0.191	0.117
github	-	-	-	-	-	-	-	-	0.329	0.549
lj	-	-	-	-	-	-	-	-	-	0.587
youtube	-	-	-	-	-	-	-	-	-	-

TABLE 6.7 – Le coefficient de Spearman des lignes de Tab.6.9 (NMI)

Tab. 6.7, NMI : Je note tout d'abord que le tuple (cora, CS) est étendu par un autre réseau de collaboration à **(cora, CS, actors)**. J'observe de plus

une corrélation inattendue : **(dblp, amazon)**. Comme observé Tab. 6.9, ces résultats sont dus à de multiples valeurs de cohérence proches de zéro. Les autres corrélations observées avec la NMI sont également observées avec la fb3.

	CS	actors	amazon	cora	dblp	flickr	football	github	lj	youtube
CS	-	-0.070	0.920	0.970	0.502	0.224	-0.434	-0.344	-0.351	-0.035
actors	-	-	-0.052	-0.157	0.472	-0.091	0.776	0.774	0.434	0.227
amazon	-	-	-	0.935	0.411	0.189	-0.455	-0.378	-0.316	-0.105
cora	-	-	-	-	0.409	0.358	-0.456	-0.381	-0.266	0.040
dblp	-	-	-	-	-	0.250	0.163	0.187	0.143	0.456
flickr	-	-	-	-	-	-	0.154	0.156	0.533	0.790
football	-	-	-	-	-	-	-	0.911	0.719	0.497
github	-	-	-	-	-	-	-	-	0.654	0.414
lj	-	-	-	-	-	-	-	-	-	0.760
youtube	-	-	-	-	-	-	-	-	-	-

TABLE 6.8 – Le coefficient de Spearman des lignes de Tab. 6.10 (fb3)

Tab. 6.8, fb3 : On note une autre corrélation surprenante Tab. 6.8 entre le réseau de co-achats et les réseaux de publication **(amazon, cora, CS)**. Ces corrélations proviennent encore une fois des valeurs de cohérence faibles observées Tab. 6.10.

Les réseaux avec une forte densité interne, **(actor, football, github)**, ont une forte corrélation interne en observant les classements des fonctions de qualité avec la fb3. Le point commun de ces réseaux est une densité élevée à l'intérieur des communautés.

J'analyse maintenant les corrélations entre les fonctions de qualité et les méthodes de comparaison. Je cherche principalement les fonctions cohérentes avec les jeux de données des contextes.

En utilisant la fb3 (Tab. 6.10), on voit qu'aucune fonction de qualité n'a de cohérence haute et consistante dans le contexte des OSN (flickr, lj, youtube). Cependant, les résultats de la NMI et de la fb3 sont anti-corrélés dans ce contexte, avec des coefficients de Spearman allant de -0.71 à -0.89.

Ainsi, Tab. 6.9 montre que la modularité a de fortes valeurs de cohérence avec les résultats de la NMI dans ce contexte. La propriété du rapport

	cc	fb3	mod	nmi	perm	sign	cond	FOMD	comp	cut_ratio	f-odf	sur
flickr	0.00	-0.71	0.75	1.00	0.61	0.07	0.61	0.07	0.14	-0.01	0.61	-0.75
lj	-0.21	-0.86	0.43	1.00	0.21	-0.18	0.50	0.25	0.32	0.35	0.50	-0.32
youtube	0.36	-0.89	0.96	1.00	0.79	0.39	0.68	0.07	0.11	0.31	0.68	0.61
cora	0.06	0.69	-0.06	1.00	0.06	-0.06	0.19	0.69	0.06	0.44	0.19	-0.06
CS	0.00	0.82	-0.25	1.00	0.00	-0.14	0.14	0.61	-0.04	0.32	0.00	-0.46
actors	-0.54	0.46	-0.89	1.00	-0.21	-0.50	-0.21	-0.21	-0.39	-0.21	-0.32	-0.57
football	0.38	0.38	0.10	1.00	0.88	0.38	-0.37	0.56	0.38	-0.87	-0.33	0.38
github	-0.29	-0.36	-0.11	1.00	-0.07	0.11	-0.11	-0.04	-0.43	-0.14	0.07	-0.04
amazon	-0.30	0.03	-0.97	1.00	-0.97	-0.12	-0.97	-0.87	0.03	-0.96	-0.97	-0.44
dblp	-0.43	0.89	-0.96	1.00	-0.96	-0.32	-0.89	-0.57	0.18	-0.88	-0.86	-0.46

TABLE 6.9 – Le coefficient de Spearman de la NMI(vérité de terrain, algorithmes) comparé aux résultats de la fonction de qualité

densité interne/externe évoquée Chap. 4 semble prévaloir pour ce contexte en utilisant la NMI. En effet, les fonctions de qualité ayant un score élevé prennent toutes cette propriété en compte (modularité, permanence, f-odf et conductance).

Les résultats des deux méthodes de comparaison sur les réseaux du contexte de la collaboration scientifique (cora, CS) sont corrélés avec la FOMD. Une FOMD élevée implique qu'un grand nombre d'individus a un degré interne élevé, elle mesure donc la propriété de densité interne. On note le score assez élevé de certaines fonctions mesurant le ratio interne/externe (cut-ratio, conductance, f-odf).

On n'observe aucune fonction de qualité cohérente avec les réseaux présentant une densité interne importante (football, github, actors) avec la NMI. Cependant, la comparaison avec la fb3 est cohérente avec la signature et de la surprise, les deux fonctions de qualité globales. Elles ont toutes deux la caractéristique de comparer la densité interne à un modèle nul.

Les deux réseaux n'appartenant pas à un contexte, amazon et dblp, ne présentent aucune corrélation satisfaisante avec les fonctions de qualité utilisées. Il est possible que les caractéristiques de ces communautés soient très éloignées de ce que l'on connaît.

J'observe aussi que la compacité est cohérente avec les résultats de la fb3 sur les réseaux à densité interne importante (football, github, actors). En plus d'un intérêt théorique, la compacité est donc expérimentalement cohérente avec un contexte.

	cc	fb3	mod	nmi	perm	sign	cond	FOMD	comp	cut_ratio	f-odf	sur
flickr	0.18	1.00	-0.21	-0.71	0.07	0.04	0.07	0.46	0.39	0.60	0.07	0.29
lj	0.29	1.00	-0.54	-0.86	-0.14	0.39	-0.46	-0.36	-0.11	-0.38	-0.46	0.32
youtube	0.04	1.00	-0.86	-0.89	-0.61	-0.07	-0.54	0.04	0.14	-0.19	-0.54	-0.32
cora	-0.64	1.00	0.04	0.69	0.29	-0.75	0.50	0.89	-0.75	0.79	0.50	-0.75
CS	-0.50	1.00	-0.14	0.82	0.14	-0.75	0.39	0.75	-0.61	0.59	0.18	-0.93
actors	0.29	1.00	-0.07	0.46	-0.79	0.43	-0.79	-0.79	0.18	-0.79	-0.64	0.36
football	1.00	1.00	0.68	0.38	0.25	1.00	-0.96	-0.05	1.00	-0.21	-0.93	1.00
github	-0.29	1.00	0.39	-0.36	-0.57	0.71	-0.61	-0.79	0.71	-0.57	-0.93	0.71
amazon	-0.86	1.00	-0.04	0.03	-0.04	-0.89	0.00	0.25	-0.93	-0.01	0.00	-0.79
dblp	-0.68	1.00	-0.79	0.89	-0.79	-0.57	-0.64	-0.32	-0.07	-0.67	-0.61	-0.71

TABLE 6.10 – Le coefficient de Spearman de la fb3(vérité de terrain, algorithmes) comparé aux résultats de la fonction de qualité

Dans ce chapitre, j'ai présenté une méthodologie permettant d'étudier la relation entre l'évaluation par les fonctions de qualité et l'évaluation par les vérités de terrain. Ces travaux ont abouti à l'identification de trois contextes qui correspondent aux sites web de réseaux sociaux, aux réseaux de collaboration scientifiques et aux réseaux à densité interne importante. Pour chacun de ces contextes, les deux méthodes de comparaison ont permis de trouver des fonctions de qualité cohérentes avec les vérités de terrain de ces contextes.

Ces travaux confirment l'hypothèse de base : les vérités de terrain peuvent être regroupées en différents contextes. Identifier ces contextes permet de catégoriser les jeux de données et de mieux comprendre leur utilisation. Cette identification ne peut se généraliser que si les chercheurs peuvent facilement analyser ces larges volumes de données. Dans le cadre de cette thèse, j'ai développé un logiciel facilitant cette démarche, le « Community Detection Algorithm Comparator » (CoDACom), présenté au prochain chapitre.

Chapitre 7

CoDACom : une boîte à outils pour la détection de communautés

CoDACom est un logiciel permettant d'appliquer une chaîne de traitements liés à la détection de communautés : algorithmes de détection de communautés, fonctions de qualité, méthodes de comparaison. Cet outil permet d'analyser séparément chaque étape du processus. Ce logiciel est facilement extensible et permet notamment la comparaison d'une nouvelle méthode avec celles déjà implémentées.

Dans ce chapitre, je présente le logiciel CoDACom (« Community Detection Algorithm Comparator »). Il regroupe une large collection d'outils pour la détection de communautés. Il peut notamment être utilisé pour :

— appliquer différents algorithmes de détection de communautés sur des graphes,

— appliquer différentes fonctions de qualité sur des partitions,

— appliquer différentes méthodes de comparaison entre les vérités de terrain et des partitions,

— évaluer la performance des algorithmes et des fonctions de qualité,

— identifier des contextes.

CoDACom est facilement extensible, un utilisateur peut donc ajouter ses propres algorithmes/fonctions/méthodes et les comparer à celles déjà implémentées. Ce logiciel a d'ailleurs évolué depuis sa conception, et certains

FIGURE 7.1 – L'interface web de CoDACom, en mode démonstration. Le chargement de fichiers sur le serveur est désactivé dans ce mode.

traitements ont été implémentés depuis la réalisation des expériences du chapitre précédent.

CoDACom a été écrit en Bourne Shell, C++ (utilisant la bibliothèque igraph [28]) et Python. Ce logiciel utilise les autotools [1] pour la compilation et la gestion des dépendances. Il est disponible, librement et gratuitement, sur `http://codacom.greyc.fr`. Ce site web décrit aussi son installation et son utilisation. CoDACom est utilisable via une interface web (Fig. 7.1) ou en ligne de commande. Je vais détailler les fonctionnalités de CoDACom en m'appuyant sur l'interface web.

7.1 Utilisation et fonctionnalités

Dans cette section je présente l'utilisation de l'interface graphique de CoDACom. Je décris ensuite comment étendre le logiciel.

Je présente les commandes générales ainsi que la partie *prétraitement* de CoDACom Fig. 7.2. Les deux premiers boutons de commande générales (1a et 1b) permettent respectivement de désélectionner et sélectionner toutes les options de l'interface, c'est-à-dire :

1. `https://www.gnu.org/software/automake/manual/html_node/Autotools-Introduction.html`

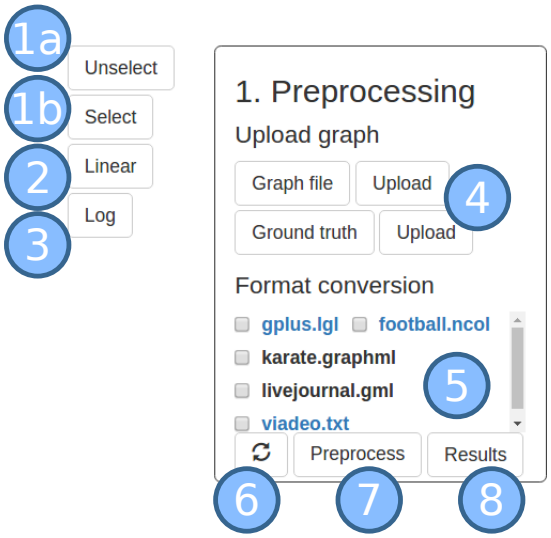


FIGURE 7.2 – Les commandes générales et la partie *prétraitement* de l'interface web de CoDACom

— les graphes à convertir (5) (Fig. 7.2),

— les algorithmes de détection de communautés à exécuter (10) (Fig. 7.10),

— les fonctions de qualité à appliquer (12) (Fig. 7.11),

— les methodes de comparaison à utiliser (15) (Fig. 7.11).

Le bouton (2) sélectionne uniquement les algorithmes de détection de communautés linéaires, ce qui peut être utile dans le cas d'un traitement sur des grands graphes. La liste des algorithmes linéaires est créée manuellement et maintenue par le gestionnaire du serveur. Le bouton (3) permet d'obtenir le fichier de log de la dernière exécution de CoDACom.

Un exemple de fichier de log est présenté Fig. 7.3. Pour des raisons de présentation, la première colonne contenant le « timestamp » de chaque évènement a été supprimée.

La partie *prétraitement* présentée Fig. 7.2 récupère et normalise les entrées à traiter. Cette partie peut être désactivée comme présenté Fig. 7.1 afin de ne pas surcharger le serveur. La zone (4) permet de charger sur le serveur les fichiers de graphe et les vérités de terrain qui leur sont associées. Les formats

```
|Starting Community Detection Algorithm Comparator (CoDACom) the 09-21-16 at 16:24:46
|The raw input graphs :
|The input graphs : /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt /www/codacom/www-prod/data/000(
|Potential method betweenness will be ignored (not in whitelist)
|Potential method conclude will be ignored (not in whitelist)
|Potential method infomap will be ignored (not in whitelist)
|Potential method leading_eigenvector will be ignored (not in whitelist)
|Potential method louvain will be ignored (not in whitelist)
|Potential method mcl will be ignored (not in whitelist)
|Potential method spinglass will be ignored (not in whitelist)
|Potential method walktrap will be ignored (not in whitelist)
|Methods that will be applied : clauset label_prop lexdfs scd
|Starting methods
|Applying methods on file karate.txt
|Executing method scd on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Successful end of the method scd on file /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Ending of method scd on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Executing method lexdfs on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Successful end of the method lexdfs on file /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Ending of method lexdfs on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Executing method label_prop on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Successful end of the method label_prop on file /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Ending of method label_prop on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Executing method clauset on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Successful end of the method clauset on file /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Ending of method clauset on input /www/codacom/www-prod/data/000001/preprocessed/inputs/karate.txt
|Applying methods on file football.txt
|Executing method scd on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Successful end of the method scd on file /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Ending of method scd on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Executing method lexdfs on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Successful end of the method lexdfs on file /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Ending of method lexdfs on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Executing method label_prop on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Successful end of the method label_prop on file /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Ending of method label_prop on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Executing method clauset on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Successful end of the method clauset on file /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Ending of method clauset on input /www/codacom/www-prod/data/000001/preprocessed/inputs/football.txt
|Ending Community Detection Algorithm Comparator (CoDACom) the 09-21-16 at 16:24:47
```

FIGURE 7.3 – Le fichier de log d'une exécution de CoDACom. Les algorithmes SCD, LexDFS, label propagation et Clauset sont appliqués à deux entrées, football et karate.

acceptés sont : graphml [2], gml [3], lgl [4] et ncol [5], et sont reconnus par l'extension du fichier. Un fichier ayant une autre extension est lu comme un fichier ncol. Tous ces formats ont une structure de données en liste d'adjacences. Pour chaque format, je fournis un exemple de deux noeuds reliés par une arête Fig. 7.4, 7.5, 7.6 et 7.7. Le format ncol étant très compact, je l'utilise comme format interne à CoDACom.

Les fichiers de vérités de terrain contiennent, pour chaque ligne, un noeud suivi des identifiants des communautés auxquelles il appartient. Un exemple de deux noeuds dont le premier appartient à deux « clusters » et qui ont un « cluster » en commun (illustré Fig. 7.8) est fourni Fig. 7.9. J'utilise aussi ce format pour représenter les partitions produites par les algorithmes de

2. http://graphml.graphdrawing.org/
3. http://www.fim.uni-passau.de/index.php?id=17297&L=1
4. http://lgl.sourceforge.net/
5. http://lgl.sourceforge.net/

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
␣␣http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="0"/>
    <node id="1"/>
    <edge source="0" target="1"/>
  </graph>
</graphml>
```

FIGURE 7.4 – Deux noeuds liés par une arête dans le format Graphml

```
graph
[
  node
  [
   id 0
  ]
  node
  [
   id 1
  ]
   edge
  [
   source 0
   target 1
  ]
]
```

FIGURE 7.5 – Deux noeuds liés par une arête dans le format GML

```
#0
1
```

FIGURE 7.6 – Deux noeuds liés par une arête dans le format LGL

```
0  1
```

FIGURE 7.7 – Deux noeuds liés par une arête dans le format ncol

détection de communautés.

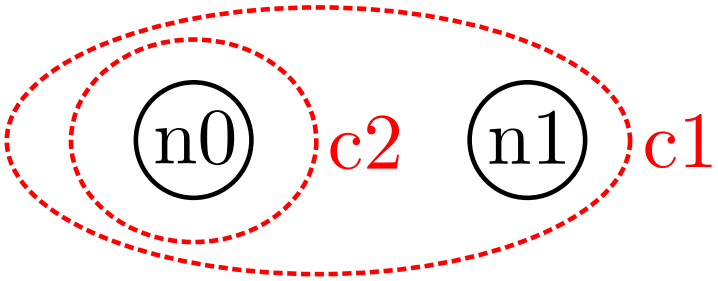


FIGURE 7.8 – Deux noeuds, n0 et n1, appartiennent respectivement aux « clusters » (c1, c2) et (c1)

```
0  1  2
1  1
```

FIGURE 7.9 – Le fichier de vérité de terrain illustré Fig. 7.8

Une fois chargés sur le serveur, les graphes apparaissent dans la zone 5. Ceux pour lesquels une vérité de terrain a été fournie sont affichés en bleu. Le bouton de rafraichissement 6 met à jour la liste dans le cas où des graphes seraient ajoutés/supprimés sans passer par l'interface. Un bouton ayant cette fonction est situé sous chacune des listes similaires de l'interface.

Le prétraitement (bouton 7) supprime les boucles (arêtes incidentes à un seul noeud) et ne conserve que la composante connexe la plus grande. L'identifiant des noeuds est modifié pour correspondre à l'ordre de visite d'un BFS à partir d'un noeud choisi uniformément aléatoirement. Ainsi, les indices des noeuds sont consécutifs, ce qui est nécessaire pour certains traitements de CoDACom. Le prétraitement est appliqué sur les fichiers sélectionnés dans la zone 5. Un fichier de correspondance faisant le lien entre les noeuds du fichier de base et le fichier prétraité est disponible via le bouton 8 en plus des fichiers prétraités eux-mêmes.

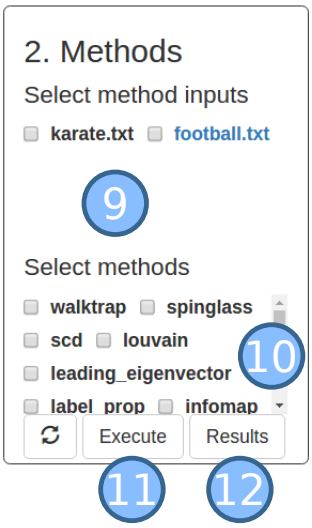


FIGURE 7.10 – La partie *traitement* de l'interface web de CoDACom

La partie *traitement* (Fig. 7.10) permet d'appliquer des algorithmes de détection de communautés aux graphes prétraités. La zone 9 liste les graphes disponibles. La zone 10 permet la sélection des algorithmes de détection de communautés à utiliser pour le traitement. CoDACom inclut les douze algorithmes de détection de communautés présentés Sec. 2.4. Le bouton 11 exécute le traitement, qui consiste à appliquer les algorithmes sélectionnés zone 10 sur les graphes sélectionnés zone 9. L'application de chaque algorithme sur chaque graphe peut être faite en parallèle, le nombre maximal de processus est configuré au niveau du serveur. Le bouton 12 permet d'obtenir les fichiers de résultat. Le format des résultat est le même que les vérités de terrain fournies en zone 4.

Les parties *analyse* et *conversion* de CoDACom sont présentées Fig. 7.11. La partie *analyse* évalue les partitions obtenues à l'étape précédente en leur appliquant des fonctions de qualité et par comparaison avec les vérités de terrain. La zone 13 affiche les trois méthodes de comparaison disponibles : NMI, F-BCubed et Omega-Index (voir Sec. 2.5). Les boutons 14 permettent de comparer les partitions fournies par les algorithmes entre elles en utilisant les méthodes sélectionnées en zone 13 et de télécharger les fichiers de résultat (un par graphe). Un exemple de fichier de résultat est fourni Fig. 7.12.

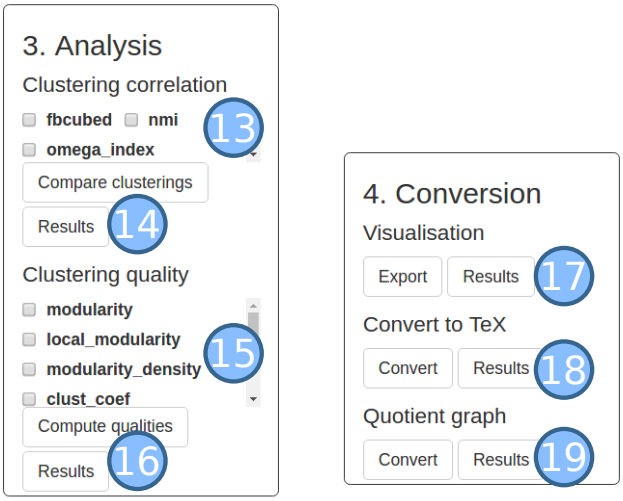


FIGURE 7.11 – Les parties *analyse* et *conversion* de l'interface web de CoDACom

```
football_ lexdfs clauset leading_eigenvector scd walktrap louvain label_prop infomap spinglass betweenness
lexdfs 1 0.649536 0.670728 0.875828 0.893209 0.859074 0.883462 0.898308 0.341398 0.92111
clauset 0.649536 1 0.521391 0.640808 0.703028 0.704157 0.750442 0.691718 0.55777 0.652335
leading_eigenvector 0.670728 0.521391 1 0.625091 0.647491 0.697669 0.647652 0.645695 0.425222 0.72358
scd 0.875828 0.640808 0.625091 1 0.885196 0.874494 0.860842 0.952641 0.272154 0.862205
walktrap 0.893209 0.703028 0.647491 0.885196 1 0.935347 0.899068 0.938695 0.33623 0.850856
louvain 0.859074 0.704157 0.697669 0.874494 0.935347 1 0.919653 0.925522 0.341573 0.915597
label_prop 0.883462 0.750442 0.647652 0.860842 0.899068 0.919653 1 0.913117 0.379643 0.895272
infomap 0.898308 0.691718 0.645695 0.952641 0.938695 0.925522 0.913117 1 0.296595 0.89951
spinglass 0.341398 0.55777 0.425222 0.272154 0.33623 0.341573 0.379643 0.296595 1 0.360741
betweenness 0.92111 0.652335 0.72358 0.862205 0.850856 0.915597 0.895272 0.89951 0.360741 1
```

FIGURE 7.12 – Un exemple de résultat du module de comparaison sur le réseau football. Chaque ligne et chaque colonne correspond à un algorithme de détection de communautés. Pour chaque paire d'algorithmes, la valeur de la méthode de comparaison est fournie (ici, la F-BCubed).

La zone 15 permet de sélectionner les fonctions de qualité à appliquer sur les partitions. CoDACom inclut les fonctions de qualité présentées Sec. 2.3. Les boutons 16 permettent :

1. d'appliquer les fonctions de qualité sélectionnées zone 15 aux partitions produites par les algorithmes,
2. de comparer les partitions aux vérités de terrain en utilisant les méthodes de comparaison sélectionnées en zone 13,
3. d'obtenir un fichier par graphe contenant les résultats des fonctions de qualité et de la comparaison avec les vérités de terrain (par exemple Fig. 7.13),

4. de comparer la cohérence entre fonctions de qualité et vérités de terrain,
5. d'obtenir un fichier de résultat par méthode de comparaison présentant cette cohérence (par exemple Fig. 7.14),
6. de mesurer la corrélation entre les résultats des différents graphes à l'étape précédente,
7. d'obtenir un fichier de résultat par méthode de comparaison présentant cette corrélation (par exemple Fig. 7.15).

```
method\quality fbcubed_gt nmi_gt modularity local_modularity permanence s_compactness
lexdfs 0.851336 0.729256 0.59077 0.928872 0.32302 251.5
clauset 0.640069 0.539896 0.559867 0.75104 0.257998 163
leading_eigenvector 0.607167 0.476078 0.492606 0.773605 0.065967 138.583
scd 0.874529 0.71853 0.57237 1.03293 0.356117 282
walktrap 0.847108 0.765333 0.602914 0.954583 0.322837 236.833
louvain 0.845974 0.763947 0.60457 0.896787 0.334029 235.167
label_prop 0.82572 0.749572 0.601959 0.871157 0.340645 261
infomap 0.891894 0.792988 0.600517 0.978762 0.366981 269
spinglass 0.279242 0.142895 0.400356 0.494916 0.284096 138
betweenness 0.8238 0.723771 0.599629 0.84634 0.33969 238.5
```

FIGURE 7.13 – Un exemple de résultat du module de qualité sur le réseau football. Les lignes correspondent à des algorithmes de détection de communautés, et les colonnes correspondent à des méthodes de comparaison ainsi qu'à des fonctions de qualité. Pour les fonctions de qualité, les valeurs sont le résultat de la fonction sur la partition générée par l'algorithme. Pour les méthodes de comparaison, les valeurs sont le résultat de la comparaison entre cette partition et la vérité de terrain.

```
file\quality fbcubed_gt local_modularity modularity nmi_gt permanence s_compactness
football 1.00 0.94 0.72 0.93 0.89 0.90
karate 1.00 -0.43 0.36 0.82 0.78 0.02
```

FIGURE 7.14 – Un exemple de résultat de cohérence entre fonctions de qualité et vérités de terrain utilisant la F-BCubed. Les lignes correspondent à des graphes et les colonnes sont les mêmes que en Fig. 7.13. Les valeurs correspondent à la cohérence entre la vérité de terrain et les fonctions (voir Chap. 6).

Je décris maintenant la partie *conversion*. Les boutons 17 génèrent et permettent de télécharger un fichier de graphe au format graphml permettant de visualiser les résultats des algorithmes. Ce fichier contient, en plus du graphe, les communautés en tant qu'étiquettes des noeuds. Les arêtes sont aussi étiquetées selon qu'elles soient internes ou externes aux communautés. Un exemple de visualisation d'un fichier de sortie de ce module est présenté

```
file\file football karate
football - 0.257
karate - -
```

FIGURE 7.15 – Corrélation entre les cohérences présentées Fig. 7.14. Les lignes et les colonnes correspondent à des graphes. Les valeurs sont le coefficient de rang de Spearman appliqué aux lignes du fichier de cohérence.

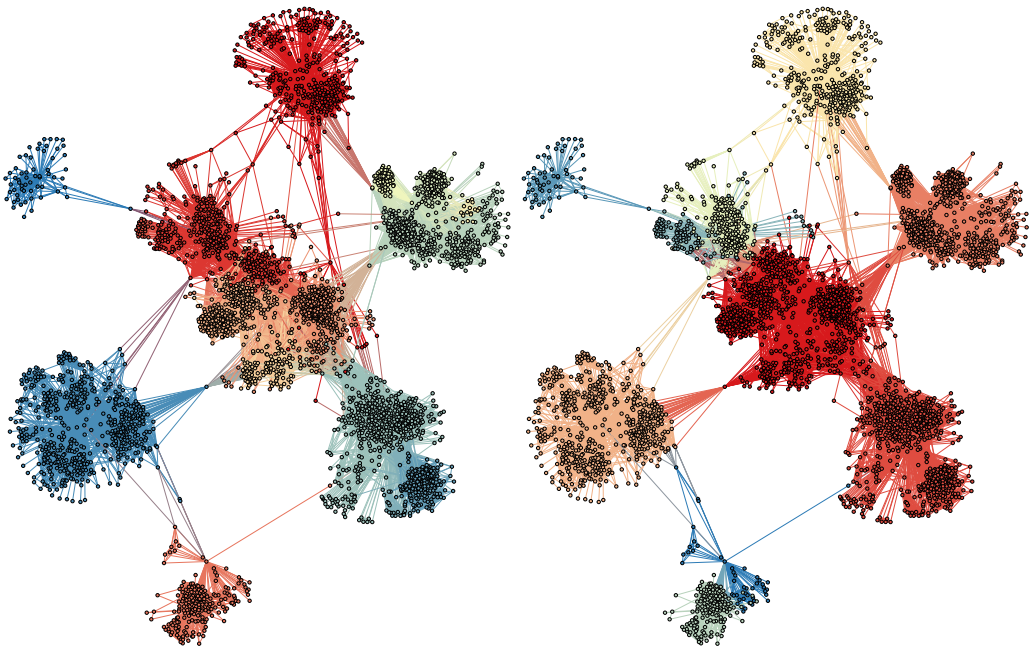


FIGURE 7.16 – Visualisations avec le logiciel Gephi(`https://gephi.org/`) d'un fichier de résultat. Il s'agit de l'algorithme de Louvain [9] (à gauche) et de MCL [94] (à droite) appliqué à un jeu de données extrait de Facebook [65].

Fig. 7.16. Le module tex (18) transforme les fichiers de résultat des analyses en tableaux LaTeXpour une intégration dans un document de ce type. Le module quotient (19) crée, pour chaque partition, le graphe quotient associé au format graphml.

CoDACom peut facilement intégrer de nouvelles implémentations d'algorithmes de détection de communautés. L'intégration d'une implémentation est réalisée grâce à un script qui a un rôle d'intermédiaire. Ce script a pour rôle d'exécuter le programme en normalisant les entrées/sorties. Ce script doit être fourni par l'utilisateur souhaitant intégrer un nouvel algorithme. Si ce script est placé dans le dossier adéquat, il est appelé au cours de l'exécution de CoDACom.

L'intégration de nouvelles fonctions de qualité et méthodes de comparaison se fait en C++. Il s'agit d'étendre la classe abstraite adéquate et de référencer la nouvelle classe dans une classe de « supervision ». Pour les méthodes de comparaison, la classe abstraite est *AbstractCM*, présentée Fig. 7.17. Pour les fonctions de qualité, il existe trois classes abstraites représentant les différents niveaux de localité (voir Sec. 2.3). Ces classes sont *AbstractQF* (« Quality Function »), *AbstractSCQF* (« Single Cluster Quality Function ») et *AbstractSVQF* (« Single Vertex Quality Function »). Ces classes peuvent être étendues de la même manière que *AbstractCM*.

```
class AbstractCM {
public:
  virtual ~AbstractCM() = 0;
  virtual double apply(std::vector<std::list<long> >& memb1,
                       std::vector<std::list<long> >& memb2)
                        = 0;
  std::string getDescription();
  std::string getName();
protected:
  std::string name;
  std::string description;
};
```

FIGURE 7.17 – L'en-tête de *AbstractCM*. Une classe fille devra redéfinir le destructur et la méthode abstraite *apply*.

7.2 Travaux connexes

Cette section présente différents outils pour la détection de communautés.

Plusieurs bibliothèques de graphes implémentent des solutions pour la détection de communautés :

— igraph [28] implémente sept algorithmes de détection de communautés, une fonction de qualité et cinq méthodes de comparaison non chevauchantes

— SNAP [6] inclut six algorithmes de détection de communautés

L'inconvénient principal de ces bibliothèques est l'utilisation du langage de programmation associé et la faible quantité de tests implémentés. Tous les

6. http://snap.stanford.edu/snap/doc.html

algorithmes implémentés dans ces bibliothèques produisant des « clusters » non chevauchants sont intégrés dans CoDACom.

Un logiciel similaire à CoDACom est Circulo [7], qui fournit quinze algorithmes de détection de communautés, une trentaine de fonctions de qualité et implémente l'Omega-Index [102] comme méthode de comparaison. Il n'inclut pas d'interface graphique.

Je note tout d'abord que les fonctions de qualité de Circulo elles proviennent toutes de l'article [103] de Yang et Leskovec. Parmi ces fonctions de qualité, la moitié de ces fonctions sont des mesures statistiques telles que la moyenne et le maximum du degré des noeuds et du coefficient de clustering local. Circulo n'inclut que des fonctions de qualité au niveau communauté.

Les algorithmes de Circulo sont implémentés et appelés en python, ce qui est une forte restriction sur le langage de programmation de tout chercheur souhaitant ajouter son implémentation au système. Par conséquent, des algorithmes connus avec des implémentations très efficaces ne sont pas inclus, comme l'implémentation de l'algorithme de Louvain [8]. Enfin, le GraphML a été choisi comme format de graphe interne alors qu'il est très redondant, ce qui entraîne des fichiers volumineux pour des graphes de grande taille.

J'en conclue que Circulo et CoDACom répondent à deux besoins différents. Circulo est utile pour obtenir un ensemble de statistiques sur divers algorithmes de détection de communautés, sur des graphes de taille restreinte (les exemples donnés vont jusqu'au million d'arêtes). Il n'effectue aucun des traitements permettant d'établir la cohérence des fonctions de qualité avec les vérités de terrain. Il ne permet donc pas d'identifier des contextes. CoDACom permet de plus d'évaluer facilement un nouvel algorithme avec des fonctions de qualité provenant de travaux divers. Certaines des implémentations inclues peuvent être exécutées sur des graphes de très grande taille, comme SCD [79] qui peut traiter un réseau incluant plus d'un milliard et demi d'arêtes sur un ordinateur de bureau.

Dans ce chapitre, j'ai présenté un logiciel nommé CoDACom. Celui-ci permet d'appliquer une chaîne de traitements liés à la détection de communautés, de la normalisation des entrées à l'analyse des résultats. Il inclut douze algorithmes de détection de communautés, douze fonctions de qualité et trois méthodes de comparaison. L'interface web permet de gérer facilement des calculs distants. CoDACom est disponible, librement et gratuitement, sur `http://codacom.greyc.fr`.

7. `http://lab41.org/circulo-a-community-detection-evaluation-framework/`
8. `https://sourceforge.net/projects/louvain/`

Chapitre 8

Conclusion

Nous avons vu que pour détecter les communautés dans un réseau social, il faut :

- identifier les propriétés structurelles que l'on estime importantes pour l'application ;
- construire un cadre d'évaluation, ce qui peut se faire de l'une ou des deux manières suivantes :
  - choisir une ou plusieurs fonctions de qualité évaluant ces propriétés structurelles ;
  - choisir les vérités de terrain qui définissent les communautés réelles ;
- sélectionner l'algorithme de détection correspondant le mieux aux besoins.

Dans cette thèse, j'ai proposé une méthodologie complète qui adresse chacun des points évoqués ci-dessus. Ma contribution à l'identification des propriétés structurelles est la définition des a-motifs. Un a-motif permet de caractériser différemment les communications internes à une communauté des communications externes. J'ai identifié que les a-motifs de type va-et-vient, spam et triangle sont caractéristiques de l'intérieur des communautés, tandis que le a-motif de chaîne est caractéristique de l'extérieur des communautés (voir Chap 3). En ce qui concerne le cadre d'évaluation, j'ai défini une nouvelle fonction de qualité : la compacité. La compacité permet d'identifier des groupes d'individus entre lesquels la diffusion d'une information est rapide (voir Chap. 4). La difficulté lors de l'évaluation par les fonctions de qualité est de déterminer si elles favorisent des communautés réelles. Dans ce but, établir la cohérence entre fonctions de qualité et vérités de terrain est crucial. J'ai proposé une méthodologie permettant de quantifier l'adéquation entre fonctions de qualité et vérités de terrain (voir Chap 6). Le logiciel

CoDACom, que j'ai écrit, assiste les praticiens dans l'application de cette méthodologie (voir Chap 7). Ce logiciel est disponible sous la licence Cecill-B à l'adresse suivante : `http://codacom.greyc.fr`. Enfin, j'ai conçu un nouvel algorithme de détection de communautés : le Lex-Clustering. Cet algorithme utilise l'algorithme de parcours de graphe LexDFS pour détecter des communautés. Le principe du LexDFS est de parcourir le graphe en favorisant les voisins des noeuds déjà visités. Le Lex-Clustering trouve des communautés qui favorisent la compacité (voir Chap. 5.

Je vais maintenant détailler chacune de ces contributions.

8.1 A-motifs

Le chapitre 3 présente de nouveaux critères pour la détection de communautés dans les réseaux temporels. Il y est montré que la façon dont les communications s'enchainent (décrite par les a-motifs) a un lien avec la structure communautaire. En effet, certains a-motifs apparaissent plus à l'intérieur qu'à l'extérieur des communautés.

J'ai montré, par l'utilisation de modèles nuls, que la causalité a une influence sur les a-motifs et leur densité interne/externe aux communautés. Or, l'agrégation des réseaux sur des fenêtres temporelles est fréquemment utilisée pour détecter les communautés sur les réseaux de communication. Cette opération détruit la causalité des communications sur ces fenêtres. L'agrégation du réseau met de côté une information qui pourrait aider à la détection de communautés. La prochaine étape de ce travail est de savoir si la causalité des communications est une information indispensable à la détection de communautés sur les réseaux de communication.

8.2 Lex-Clustering et compacité

Le chapitre 4 présente la définition d'une nouvelle fonction de qualité : la compacité. Celle-ci se fonde sur un modèle de diffusion simple de l'information dans un groupe, l'« Independant Cascade Model » [40, 53] avec une probabilité de transmission de 1. La compacité mesure la vitesse de propagation de cette diffusion. Dans le chapitre 4, je compare la compacité à d'autres fonctions de qualité. J'y montre que la compacité pénalise les structures où des individus sont peu connectés au reste du groupe. Je montre aussi que la compacité satisfait les axiomes de Van Laarhoven et Marchiori [95].

L'inconvénient de cette fonction de qualité est son temps de calcul. Il est donc déraisonnable de créer un algorithme de détection de communautés basé

sur le calcul explicite de la compacité. En pratique, j'utilise une approximation du diamètre pour pouvoir calculer la compacité. J'ai conçu la compacité en tant que fonction de qualité car, selon moi, la vitesse de la diffusion d'information dans une communauté est sa caractéristique principale.

Le chapitre 5 présente un nouvel algorithme hiérarchique de détection de communautés basé sur l'algorithme du LexDFS : le Lex-Clustering. Il se base sur un algorithme de parcours de graphe visitant en priorité les noeuds voisins des noeuds visités récemment. C'est par l'exécution répétée de ce parcours que les arêtes intra-communautaires sont détectées. J'utilise ensuite un algorithme hiérarchique pour détecter les communautés à partir de ces arêtes. Le Lex-Clustering produit des « clusters » denses, comme montré par la comparaison avec des vérités de terrain.Cet algorithme produit des « clusters » favorisés par la compacité.

8.3 Fonctions de qualité et vérités de terrain

Le chapitre 6 présente une méthodologie pour établir la cohérence entre les fonctions de qualité et les vérités de terrain.

J'observe que certaines fonctions de qualité évaluent les partitions de manière cohérente avec certaines vérités de terrain. De plus, il existe des ensembles de vérités de terrain qui présentent des résultats semblables quand on compare entre elles leur cohérence avec les fonctions de qualité. J'appelle ces ensembles de vérités de terrain des contextes. Une fonction de qualité peut être cohérente avec les vérités de terrain d'un contexte, mais incohérente avec ceux d'un autre. Identifier le contexte d'un réseau sans vérité de terrain permet de connaître la fonction de qualité à utiliser pour évaluer les différents algorithmes de détection de communautés.

Notons que, même si l'ensemble des données utilisées par ces expériences est de taille importante, il est impossible de savoir si d'autres contextes existent dans d'autres vérités de terrain sans les avoir analysés. Il serait donc intéressant d'enrichir ces résultats par l'analyse de vérités de terrain supplémentaires.

Il s'agit de l'un des objectifs du logiciel CoDACom, présenté au chapitre 7. Ce logiciel est aussi adapté à divers besoins en détection de communautés grâce à la bibliothèque d'outils qu'il contient. Il inclut notamment une suite de tests permettant l'analyse d'algorithmes de détection de communautés, de fonctions de qualité et de vérités de terrain. Il est de plus facilement extensible grâce à sa conception modulaire. Il contient aussi une interface web permettant de le manipuler. CoDACom est disponible à l'adresse `http://codacom.greyc.fr/`.

Annexe A

Démarches et description du réseau d'e-mails de l'université de Caen

La première démarche fut une étude de faisabilité, épaulée par la Direction des Systèmes d'Information (DSI). Il s'est avéré que les métadonnées des e-mails émis et reçus par des adresses gérées par l'université étaient facilement récupérables par le biais de logs. De plus, un service d'adresses LDAP (« Lightweight Directory Access Protocol ») contient des métadonnées sur les individus utilisant les adresses en question.

La question légale s'est alors posée, la protection des données privées informatiques étant gérée par la Commission Nationale de l'Informatique et des Libertés (CNIL). J'ai donc identifié les métadonnées intéressantes pour l'étude, et ai contacté l'interlocuteur privilégié pour les questions relatives à la CNIL, le Correspondant Informatique et Libertées (CIL).

Le principe de base est de rendre la levée de l'anonymat extrêmement difficile par un individu isolé. Il a donc fallu revoir certaines méta-données pour qu'elles ne permettent pas une identification trop aisée des individus impliqués. Par exemple, certains laboratoires sont de petite taille (inférieure à cinq). La liste des membres de ce laboratoire est, le plus souvent, disponible facilement en ligne. Si l'âge ou le sexe est une méta-donnée fournie, il devient aisé d'obtenir les méta-données des communications d'un membre du laboratoire en particulier. C'est pourquoi nous avons, par exemple, décidé d'exclure l'âge des méta-données extraites pour le personnel, et ne fournir que des tranches d'âges de deux ans pour les étudiants, avec deux tranches extrêmales (moins de 18 ans et plus de 30 ans).

Ensuite, un membre de la DSI a effectué les traitements d'extraction, de normalisation et d'anonymisation des données. Ceux-ci ont étés effectués

sur des serveurs sécurisés et les clefs de correspondance entre les données non anonymisées et anonymisées ont étés détruites après traitement. La procédure entière a été présentée au CIL, qui a validé le traitement d'anonymisation et la mise à disposition du jeu de données à la communauté scientifique.

Afin d'avoir au mieux possible une bijection entre les utilisateurs et les adresses, les traitements effectués à partir du réseau brut sont les suivants :

— suppression des adresses invalides (fautes de frappe dans le destinataire, ...)

— réunion des alias internes

Les données s'étalent sur une période de trois mois durant l'année scolaire 2015-2016. Les métadonnées collectées sur les e-mails sont les champs FROM, TO et DATE, c'est-à-dire les informations nécessaires pour le flux de liens. Pour les utilisateurs actifs durant cette période, les métadonnées suivantes sont rassemblées :

— type : si l'utilisateur est une personne ou une liste de diffusion

— catégorie : la catégorie socioprofessionnelle (employé ou étudiant)

— sexe

— âge : seulement pour les étudiants, par tranche de deux ans entre 18 et 30 ans

— l'entité administrative de rattachement, comme l'Unité de Formation et de Recherche (UFR) ou un Institut Universitaire Technologique (IUT)

— laboratoire

— section du Conseil National des Universités (CNU) : la discipline que pratique l'individu (informatique, biologie, etc.)

Bibliographie

[1] Rodrigo Aldecoa and Ignacio Marín. Surprise maximization reveals the community structure of complex networks. *Scientific Reports*, 3, January 2013.

[2] Hélio Almeida, Dorgival Guedes, Wagner Meira Jr, and Mohammed J. Zaki. Is there a best quality metric for graph clusters ? In *Machine Learning and Knowledge Discovery in Databases*, pages 44–59. Springer, 2011.

[3] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4) :461–486, 2009.

[4] Alex Arenas, Albert Díaz-Guilera, and Conrad J. Pérez-Vicente. Synchronization reveals topological scales in complex networks. *Physical review letters*, 96(11) :114102, 2006.

[5] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th International Conference on Computational Linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics, 1998.

[6] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. Everyone's an influencer : quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74. ACM, 2011.

[7] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528. ACM, 2012.

[8] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439) :509–512, 1999.

[9] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large net-

works. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10) :P10008, 2008.

[10] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. *On modularity – NP-Completeness and beyond*. Citeseer, 2006.

[11] Julián Candia, Marta C. González, Pu Wang, Timothy Schoenharl, Greg Madey, and Albert-László Barabási. Uncovering individual and collective human dynamics from mobile phone records. *Journal of Physics A : Mathematical and Theoretical*, 41(22) :224015, 2008.

[12] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 197–210, 2012.

[13] Ciro Cattuto, Wouter Van den Broeck, Alain Barrat, Vittoria Colizza, Jean-François Pinton, and Alessandro Vespignani. Dynamics of Person-to-Person Interactions from Distributed RFID Sensor Networks. *PLoS ONE*, 5(7) :e11596, July 2010.

[14] Remy Cazabet, Frederic Amblard, and Chihab Hanachi. Detection of overlapping communities in dynamical social networks. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 309–314. IEEE, 2010.

[15] D. Centola. The Spread of Behavior in an Online Social Network Experiment. *Science*, 329(5996) :1194–1197, September 2010.

[16] Tamal Chakraborty, Sujit Sikdar, Vihar Tammana, Niloy Ganguly, and Arjun Mukherjee. Computer science fields as ground-truth communities : Their impact, rise and fall. In *Proceedings of Advances in Social Networks Analysis and Mining (ASONAM), 2013*, pages 426–433. IEEE, 2013.

[17] Tanmoy Chakraborty, Sandipan Sikdar, Niloy Ganguly, and Animesh Mukherjee. Citation interactions among computer science fields : a quantitative route to the rise and fall of scientific research. *Social Network Analysis and Mining*, 4(1) :1–18, 2014.

[18] Tanmoy Chakraborty, Sriram Srinivasan, Niloy Ganguly, Animesh Mukherjee, and Sanjukta Bhowmick. On the permanence of vertices in network communities. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 2014, pages 1396–1405, New York, NY, USA, 2014. ACM.

[19] Mingming Chen, Tommy Nguyen, and Boleslaw K. Szymanski. A new metric for quality of network community structure. *arXiv preprint arXiv :1507.04308*, 2015.

[20] Aaron Clauset, M. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), December 2004.

[21] Linda M. Collins and Clyde W. Dent. Omega : A General Formulation of the Rand Index of Cluster Recovery Suitable for Non-disjoint Solutions. *Multivariate Behavioral Research*, 23(2) :231–242, April 1988.

[22] Derek G. Corneil, Barnaby Dalton, and Michel Habib. LDFS-Based Certifying Algorithm for the Minimum Path Cover Problem on Cocomparability Graphs. *SIAM Journal on Computing*, 42(3) :792–807, January 2013.

[23] Derek G. Corneil and Richard M. Krueger. A unified view of graph searching. *SIAM Journal on Discrete Mathematics*, 22(4) :1259–1276, January 2008.

[24] Robin Cowan and Nicolas Jonard. Network structure and the diffusion of knowledge. *Journal of Economic Dynamics and Control*, 28(8) :1557–1575, June 2004.

[25] Jean Creusefond, Thomas Largillier, and Sylvain Peyronnet. Finding compact communities in large graphs. In *Proceedings of Advances in Social Networks Analysis and Mining (ASONAM), 2015*, pages 1457–1464. ACM, 2015.

[26] Jean Creusefond, Thomas Largillier, and Sylvain Peyronnet. A lexdfs-based approach on finding compact communities. In *From Social Data Mining and Analysis to Prediction and Community Detection*. Springer, 2016.

[27] Jean Creusefond, Thomas Largillier, and Sylvain Peyronnet. On the evaluation potential of quality functions in community detection for different contexts. In *Advances in Network Science*, pages 111–125. Springer, 2016.

[28] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems :1695, 2006.

[29] Munmun De Choudhury, Hari Sundaram, Ajita John, and Dorée Duncan Seligmann. Social synchrony : Predicting mimicry of user actions in online social media. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 151–158. IEEE, 2009.

[30] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. Mixing local and global information for community detection in large networks. *Journal of Computer and System Sciences*, 80(1) :72–87, February 2014.

[31] David Easley and Jon Kleinberg. *Networks, crowds, and markets : Reasoning about a highly connected world*. Cambridge University Press, 2010.

[32] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7) :1575–1584, April 2002.

[33] Emilio Ferrara, Pasquale De Meo, Salvatore Catanese, and Giacomo Fiumara. Detecting criminal organizations in mobile phone networks. *Expert Systems with Applications*, 41(13) :5733–5750, 2014.

[34] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of Web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160. ACM Press, 2000.

[35] G.W. Flake, S. Lawrence, C.L. Giles, and F.M. Coetzee. Self-organization and identification of Web communities. *Computer*, 35(3) :66–70, March 2002.

[36] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5) :75–174, February 2010.

[37] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1) :36–41, 2007.

[38] Noé Gaumont, Tiphaine Viard, Raphaël Fournier-S'niehotta, Qinna Wang, and Matthieu Latapy. Analysis of the temporal and structural features of threads in a mailing-list. In *Complex Networks VII*, pages 107–118. Springer, 2016.

[39] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12) :7821–7826, 2002.

[40] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the Network : A Complex Systems Look at the Underlying Process of Word-of-Mouth. *Marketing Letters*, 12(3) :211–223, 2001.

[41] Mark S Granovetter. The strength of weak ties. *American journal of sociology*, pages 1360–1380, 1973.

[42] Roger Guimerà, Marta Sales-Pardo, and Luís A. Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2), August 2004.

[43] Mika Gustafsson, Michael Hörnquist, and Anna Lombardi. Comparison and validation of community structures in complex networks. *Physica A : Statistical Mechanics and its Applications*, 367 :559–576, 2006.

[44] Vicenç Gómez, Andreas Kaltenbrunner, and Vicente López. Statistical analysis of the social network and discussion threads in slashdot. In *Proceedings of the 17th international conference on World Wide Web*, pages 645–654. ACM, 2008.

[45] Matthew B. Hastings. Community detection as an inference problem. *Physical Review E*, 74(3) :035102, 2006.

[46] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301) :13–30, 1963.

[47] Petter Holme. Modern temporal network theory : a colloquium. *The European Physical Journal B*, 88(9), September 2015.

[48] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3) :97–125, 2012.

[49] Darko Hric, Richard K. Darst, and Santo Fortunato. Community detection in networks : Structural communities versus ground truth. *Physical Review E*, 90(6), December 2014.

[50] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering : a review. *ACM computing surveys (CSUR)*, 31(3) :264–323, 1999.

[51] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings : Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3) :497–515, 2004.

[52] Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1), January 2011.

[53] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. pages 137–146, 2003.

[54] Mikko Kivelä, Raj Kumar Pan, Kimmo Kaski, János Kertész, Jari Saramäki, and Márton Karsai. Multiscale analysis of spreading in a large communication network. *Journal of Statistical Mechanics : Theory and Experiment*, 2012(03) :P03005, 2012.

[55] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS*, 2004.

[56] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. Temporal motifs. In *Temporal Networks*, Understanding Complex Systems. Springer, Berlin, Heidelberg, 2013.

[57] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1) :79–86, 1951.

[58] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3) :033015, 2009.

[59] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), October 2008.

[60] Thomas Largillier, Guillaume Peyronnet, and Sylvain Peyronnet. SpotRank : a robust voting system for social news websites. page 59. ACM Press, 2010.

[61] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. page 915. ACM Press, 2008.

[62] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution : Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1) :2, March 2007.

[63] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704. ACM, 2008.

[64] Jure Leskovec, Kevin J. Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[65] Jure Leskovec and Julian J. Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[66] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Facetnet : a framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 685–694. ACM, 2008.

[67] Clémence Magnien, Matthieu Latapy, and Michel Habib. Fast computation of empirically tight bounds for the diameter of massive graphs. *Journal of Experimental Algorithmics*, 13 :1.10, February 2009.

[68] Miller McPhersonl, Lynn Smith-Lovinl, and James M. Cook. BIRDSOF A FEATHER : Homophily. *Annual review of sociology*, 27 :415–444, 2001.

[69] Radosław Michalski, Sebastian Palus, and Przemysław Kazienko. Matching Organizational Structure and Social Network Extracted from Email Communication. In Wil van der Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, and Witold Abramowicz, editors, *Business Information Systems*, volume 87, pages 197–206. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[70] Stanley Milgram. The small world problem. *Psychology today*, 2(1) :60–67, 1967.

[71] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC 2007)*, San Diego, CA, 2007.

[72] Marija Mitrović and Bosiljka Tadić. Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities. *Physical Review E*, 80(2) :026123, 2009.

[73] Stefanie Muff, Francesco Rao, and Amedeo Caflisch. Local modularity measure for network clusterizations. *Physical Review E*, 72(5), November 2005.

[74] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), September 2006.

[75] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2) :026113, 2004.

[76] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2) :155–163, May 2009.

[77] Leto Peel, Daniel B. Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *arXiv preprint arXiv :1608.05878*, 2016.

[78] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10(2) :191–218, 2006.

[79] Arnau Prat-Pérez, David Dominguez-Sal, and Josep-Lluis Larriba-Pey. High quality, scalable and parallel community detection for large real graphs. pages 225–236. ACM Press, 2014.

[80] Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2) :88–91, 1994.

[81] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9) :2658–2663, 2004.

[82] Usha Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), 2007.

[83] Matthew J. Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *Proceedings of the 24th international conference on Machine learning*, pages 783–790. ACM, 2007.

[84] P. Krishna Reddy, Masaru Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. In *International Workshop on Databases in Networked Information Systems*, pages 188–200. Springer, 2002.

[85] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1), July 2006.

[86] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4) :1118–1123, 2008.

[87] Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3) :269–287, September 1983.

[88] Lovro Šubelj and Marko Bajec. Model of complex networks based on citation dynamics. In *Proceedings of the 22nd international conference on World Wide Web*, pages 527–530, 2013.

[89] Lionel Tabourier, Alina Stoica, and Fernando Peruani. How to detect causality effects on large dynamical communication networks : a case study. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference On*, pages 1–7. IEEE, 2012.

[90] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, 22(2) :215–225, 1975.

[91] V. A. Traag, R. Aldecoa, and J.-C. Delvenne. Detecting communities using asymptotical surprise. *Physical Review E*, 92(2), August 2015.

[92] V. A. Traag, G. Krings, and P. Van Dooren. Significant Scales in Community Structure. *Scientific Reports*, 3, October 2013.

[93] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of Facebook networks. *Physica A : Statistical Mechanics and its Applications*, 391(16) :4165–4180, August 2012.

[94] Stijn van Dongen. *Graph clustering by flow simulation.* PhD thesis, 2000.

[95] Twan Van Laarhoven and Elena Marchiori. Axioms for graph clustering quality functions. *The Journal of Machine Learning Research*, 15(1) :193–215, 2014.

[96] Jordan Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609 :245–252, 2016.

[97] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42. ACM, 2009.

[98] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393(6684) :440–442, 1998.

[99] Yen-Chuen Wei and Chung-Kuan Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, pages 298–301. IEEE, 1989.

[100] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. Virality Prediction and Community Structure in Social Networks. *Scientific Reports*, 3, August 2013.

[101] F. Y. Wu. The Potts model. *Reviews of Modern Physics*, 54(1) :235–268, January 1982.

[102] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping community detection in networks : The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4) :1–35, August 2013.

[103] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1) :181–213, 2012.

[104] Yi-Qing Zhang, Xiang Li, Jian Xu, and Athanasios Vasilakos. Human Interactive Patterns in Temporal Networks. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 45(2) :214–222, February 2015.

[105] Mesut Yucel, Lev Muchnik, and Uri Hershberg. Detection of network communities with memory-biased random walk algorithms. *Journal of Complex Networks*, page cnw007, April 2016.

[106] Qiankun Zhao, Yuan Tian, Qi He, Nuria Oliver, Ruoming Jin, and Wang-Chien Lee. Communication motifs : a tool to characterize social

communications. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1645–1648. ACM, 2010.

[107] Haijun Zhou and Reinhard Lipowsky. Network brownian motion : A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In *International conference on computational science*, pages 1062–1069. Springer, 2004.

Caractériser et détecter les communautés dans les réseaux sociaux

Résumé : Dans cette thèse, je commence par présenter une nouvelle caractérisation des communautés à partir d'un réseau de messages inscrits dans le temps. Je montre que la structure de ce réseau a un lien avec les communautés : on trouve majoritairement des échanges d'information à l'intérieur des communautés tandis que les frontières servent à la diffusion.
Je propose ensuite d'évaluer les communautés par la vitesse de propagation des communications qui s'y déroulent avec une nouvelle fonction de qualité : la compacité. J'y présente aussi un algorithme de détection de communautés, le Lex-Clustering, basé sur un algorithme de parcours de graphe qui reproduit des caractéristiques des modèles de diffusion d'information.
Enfin, je présente une méthodologie permettant de faire le lien entre les fonctions de qualité et les vérités de terrain. J'introduis le concept de contexte, des ensembles de vérités de terrain qui présentente des ressemblances. Je mets à disposition un logiciel nommé CoDACom (Community Detection Algorithm Comparator, codacom.greyc.fr) permettant d'appliquer cette méthodologie ainsi que d'utiliser un grand nombre d'outils de détection de communautés.

Mots-clefs : détection de communautés ; vérité de terrain ; graphe ; algorithme

Caracterising and detecting communities in social networks

Abstract : In this thesis, I first present a new way of characterising communities from a network of timestamped messages. I show that its structure is linked with communities : communication structures are over-represented inside communities while diffusion structures appear mainly on the boundaries.
Then, I propose to evaluate communities with a new quality function, compacity, that measures the propagation speed of communications in communities. I also present the Lex-Clustering, a new community detection algorithm based on the LexDFS graph traversal that features some characteristics of information diffusion.
Finally, I present a methodology that I used to link quality functions and ground-truths. I introduce the concept of contexts, sets of ground-truths that are similar in some way. I implemented this methodology in a software called CoDACom (Community Detection Algorithm Comparator, codacom.greyc.fr) that also provides many community detection tools.

Keywords : community detection ; ground-truth ; graph ; algorithm

Discipline : Informatique et applications
Laboratoire : Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (GREYC - UMR 6072)